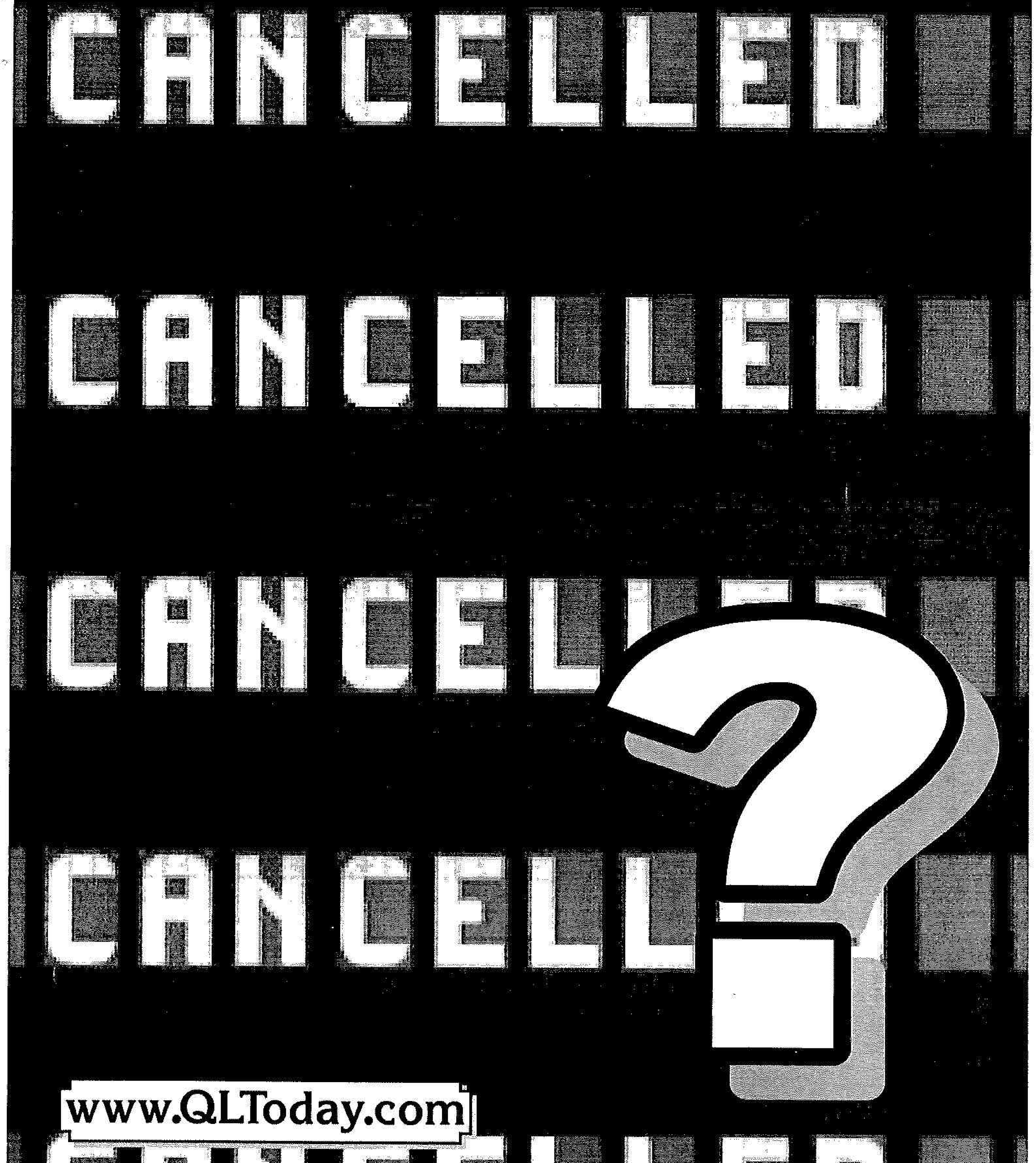
QL Today

Volume 17
Issue 3
March - May
2013

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...



www.QLToday.com

Contents

Advertisers
in alphabetical order

QL *Today*

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software	Tel. +49 203 502011
Kaiser-Wilhelm-Str. 302	Fax +49 203 502012
47169 Duisburg	email: smsq@j-m-s.com
Germany	email: QLToday@j-m-s.com

Editor:

Geoff Wicks	Tel. +44 1332 271366
Flat 5b	email: gtwicks@btinternet.com
Wordsworth Avenue	email: QLToday@j-m-s.com
Derby DE24 9HQ	
United Kingdom	

Co-Editor & UK Office:

Bruce Nicholls	Tel +44 20 71930539
38 Derham Gardens	Fax +44 870 0568755
Upminster	email: qltoday@q-v-d.demon.co.uk
Essex RM14 3HA	email: QLToday@j-m-s.com
United Kingdom	

QL *Today* is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at http://www.dilwyn.uk6.net/arch/index.html

The deadline for the next issue is the 31st of May 2013!

Editorial

by Geoff Wicks

Three news stories have dominated the QL scene this year, one of which you will be reading for the first time in this issue of QL Today. Volume 17 will be our final volume. Issue 4 will not be published in the Summer, but will be postponed until the Autumn. This is to give us time to scan volumes 16 and 17 so that readers can have a complete electronic archive of every issue of the magazine.

Another of the news stories is a sad one, although not unexpected. In this issue we report the death of Bill Richardson, one of the unforgettable characters of the QL community. Most of us have our own memories of him. Bill once gave me a minor rebuke for something I wrote in the very first issue of QL Today. I had suggested that most QL software authors do not charge for their labour, and, ever the trader, Bill told me that I should.

The third news item is more pleasurable with the release of a new QL product, SMSQmulator. It has excited much interest in the QL community with a continuing discussion on the QL-users email group. Indeed it brought out the best in the group and showed how much we need it. For several days, if not weeks, there was instant feedback on the program including bug reports. These were quickly followed up by SMSQmulator author Wolfgang Lenerz, who was constantly updating the program. Wolfgang has given the QL community a morale boost and shown that there is still a great deal of life in the QL.

To return to QL Today. When Jochen phoned me with the proposal to make this the last volume of QL Today, I had little difficulty in agreeing. Although we have been blessed with some very faithful writers for whom we are grateful, they were, unfortunately, far too few in number. I had a constant nightmare of losing just one writer and then being unable to fill the magazine.

When the readers pay their subscription, we have a duty to produce a quality product. Without more writers I could not guarantee continuing to do so. Like Jochen I would rather close the magazine while it is still relatively strong than struggle on before finally letting our readers down.

To our present writers I would like to say a special thank you. During this volume you have often provided your copy well before the deadline date. This took a lot of pressure off Jochen and myself as we started work on each new issue.

Let us not be downhearted by the closure of QL Today. There is a long QL tradition of new initiatives being started when something closes down. QL Today itself rose from the ashes of IQLR, and has survived longer than any other QL publication with the exception of the Quanta Magazine.

The future will not, however, lie in a printed magazine. The economics are against it. For example, producing the Quanta Magazine took up half of Quanta's income last year and almost two thirds the year before. It was one reason Quanta had to raise its subscription and introduce a postal supplement for UK members.

The future could lie in some form of web initiative. There is and will remain a need for more in depth QL articles and information than is possible in either the QL-users email group or the QL forum.

There can be life after QL Today. The challenge is there. Who will take it up?

NEW JAVA EMULATOR

The New Year brought with it a new SMSQ/E emulator which *Wolfgang Lenerz* released towards the end of January. SMSQmulator is a Java emulation of SMSQ/E and raises the possibility of some form of QL platform on portable devices. The emulator runs at about one tenth of the speed of QPC2 for multicore processors, but much slower on single core systems.

The emulator dominated discussion on the QL users email group for about 3 weeks after its release with contributors praising the quality of the product and reporting very few bugs and program incompatibilities. Wolfgang has quickly responded to any bug reports and since its first release there have been several updates posted on his website.

SMSQmulator is intended purely as an SMSQ/E emulator and thus is not compatible with programs that can only be run on native hardware.

The emulator can be downloaded from:
www.wlenerz.com/SMSQmulator
The web address is case sensitive.

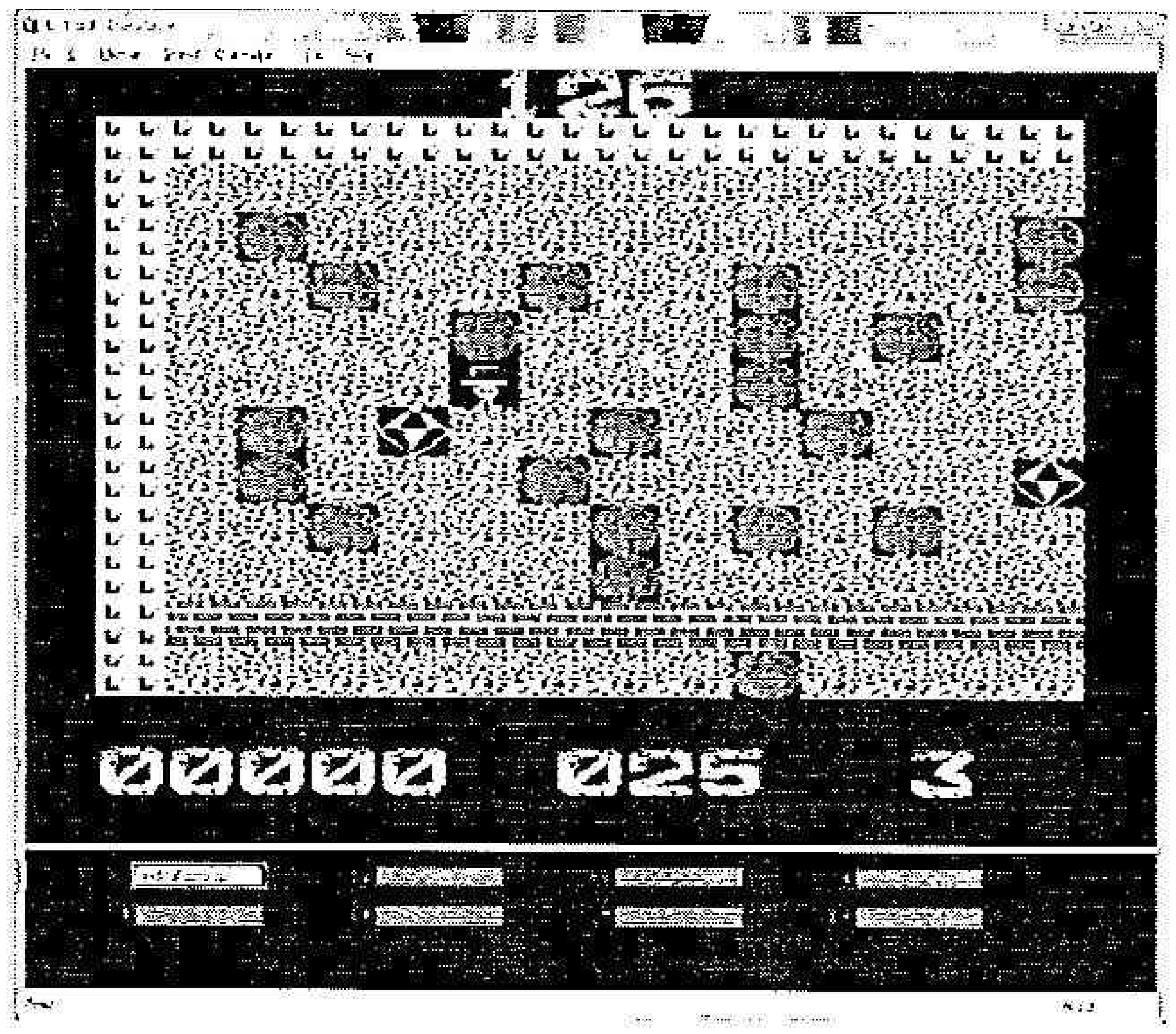
Dilwyn Jones' Updates

Dilwyn Jones has a long list of updates and new products that can be downloaded from his site:

ROCKFALL and MANDELSPEED NOW FREEWARE

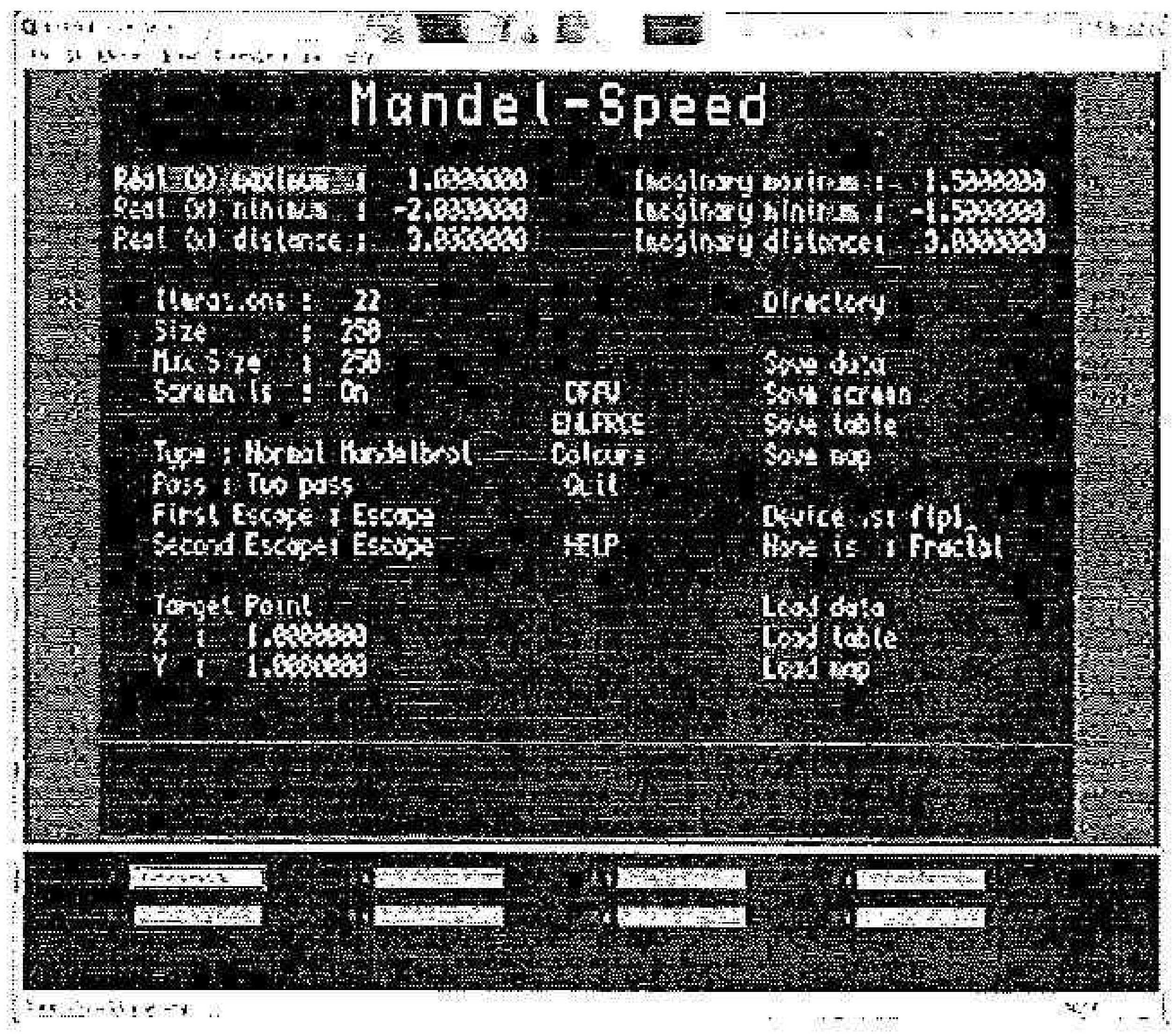
Originally written by Andy Toone, this colourful Boulderdash style arcade game from the 1980s is now available as freeware after Rich Mellor secured permission from the author. I have made it available for download in two versions, one for QemuLator which can be run direct from the zip file by linking the zip file to one of the QemuLator drive slots (the zip file contains a BOOT program which should allow it to start automatically from the zip file when the emulation is started) and a second version without QemuLator file headers, which prevented it running on other QL systems. Note that although it seemed to run fine on Minerva ROM with QemuLator, I was not able to get it to run in SMSQ/E on QPC2. In the game, you have to move around through the dirt collecting diamonds while avoiding falling boulders and monsters which chase you.

Download this game from
http://www.dilwyn.me.uk/games/index.html



A Mandelbrot graphics plot tool called Mandelspeed from the same author is also now freeware and comes in separate versions for Q-emuLator and other QL systems.

Download both versions from
http://www.dilwyn.me.uk/graphics/index.html



```
                     Mandel-Speed

Real (x) maximum :  1.0000000     Imaginary maximum :  1.5000000
Real (x) minimum : -2.0000000     Imaginary minimum : -1.5000000
Real (x) distance:  3.0000000     Imaginary distance:  3.0000000

  Iterations :  22                            Directory
  Size       :  250
  Max Size   :  250                           Save data
  Screen is  :  On               OFFU         Save screen
                                 BILFRCE      Save table
  Type : Normal Mandelbrot       Colours      Save map
  Pass : Two pass                Quit
  First Escape : Escape                       Device is: flp1_
  Second Escape: Escape          HELP         Name is  : Fractal

  Target Point                                Load data
  X :  1.0000000                              Load table
  Y :  1.0000000                              Load map
```

PCB DESIGN UPDATE

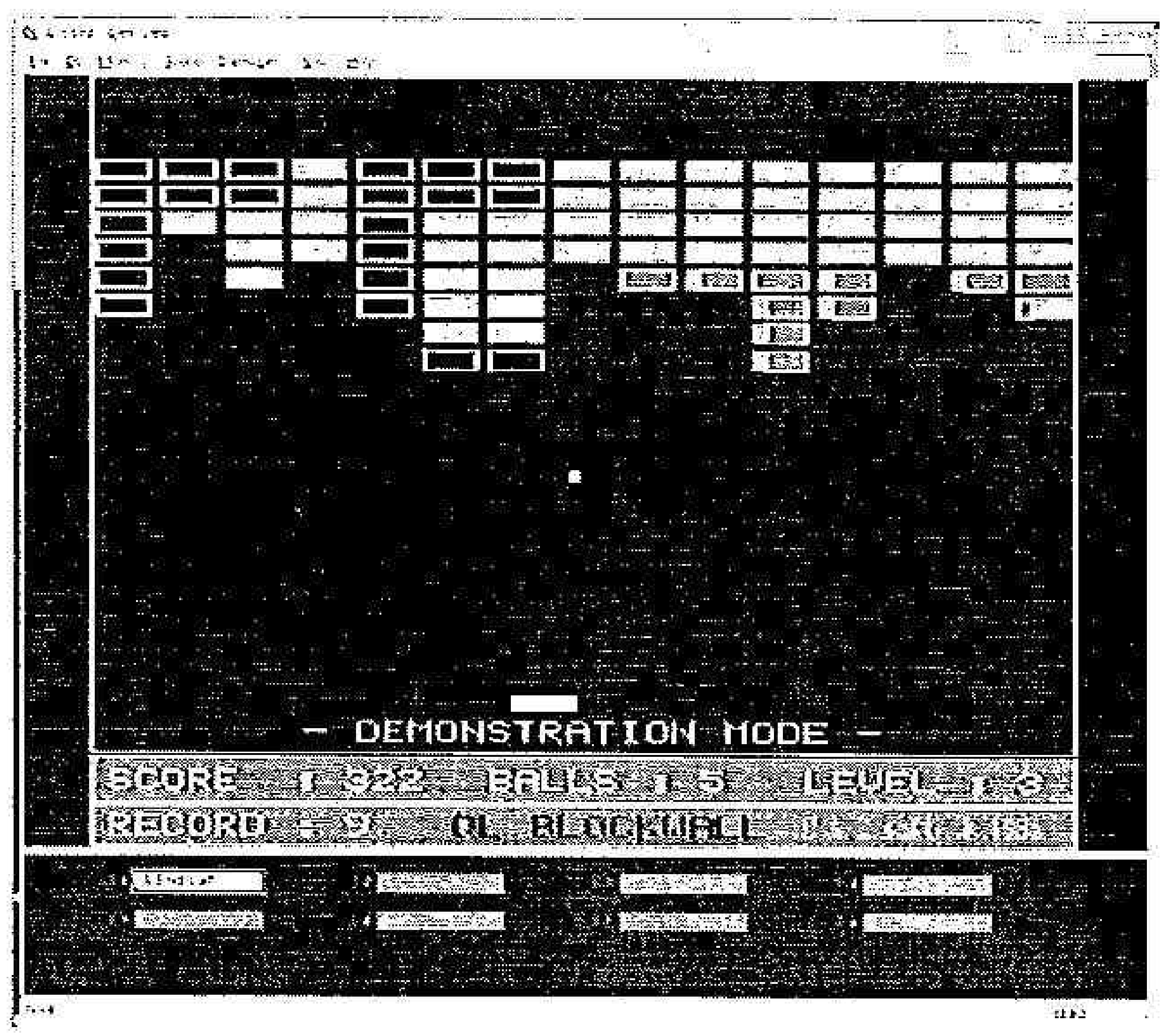
Malcolm Lear has released a further update to his PCB designs software. Version 7.25 updates include, according to Malcolm: 'The biggest improvement with this version is the editing and viewing of the NC drill sizes. 'h' toggles the screen view on/off and 'Ctrl h' starts up a drill editor.'

Download this latest update from the Graphics

page on my website at
http://www.dilwyn.me.uk/graphics/index.html

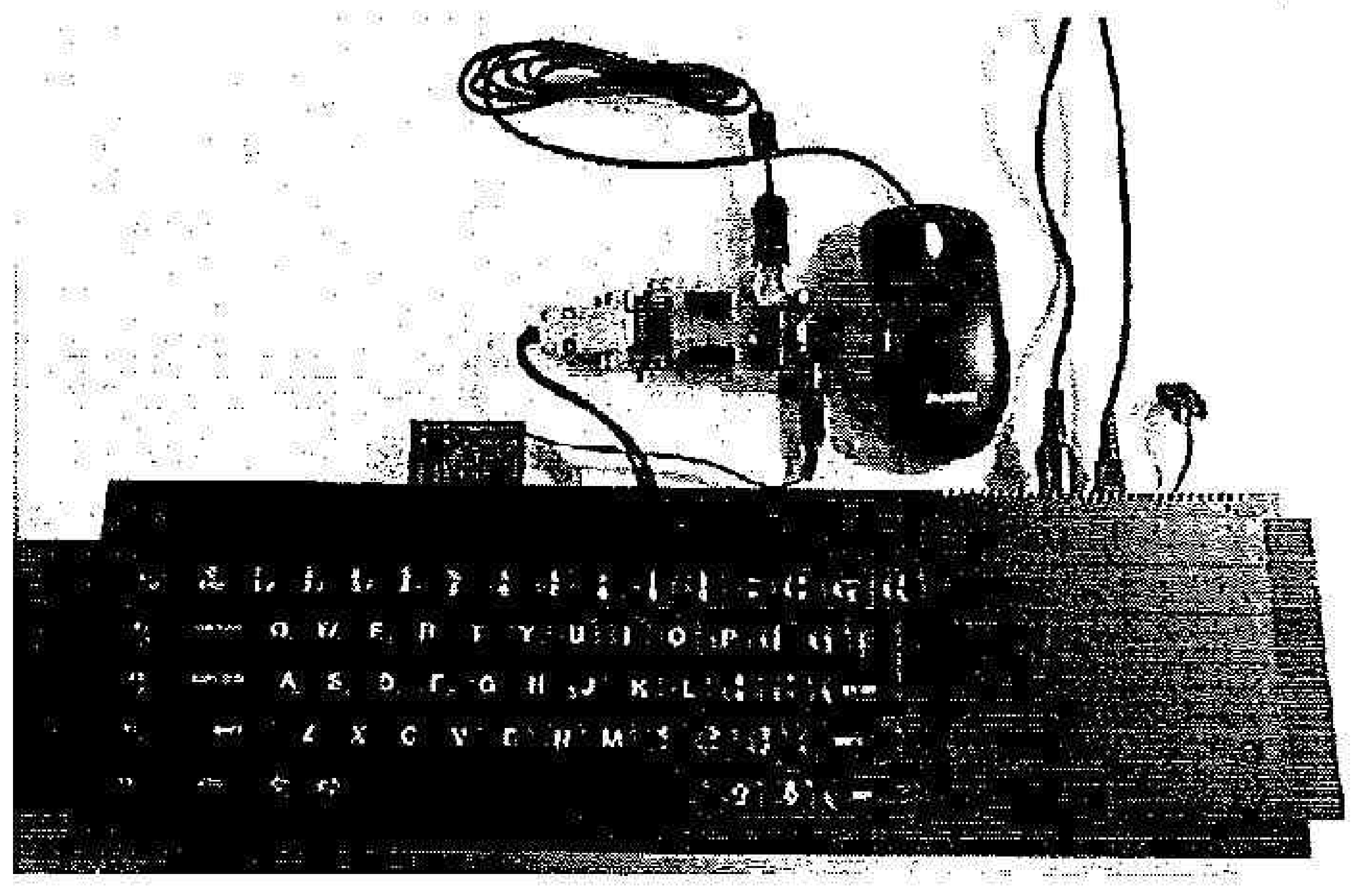
PROJECT GC QL MOUSE

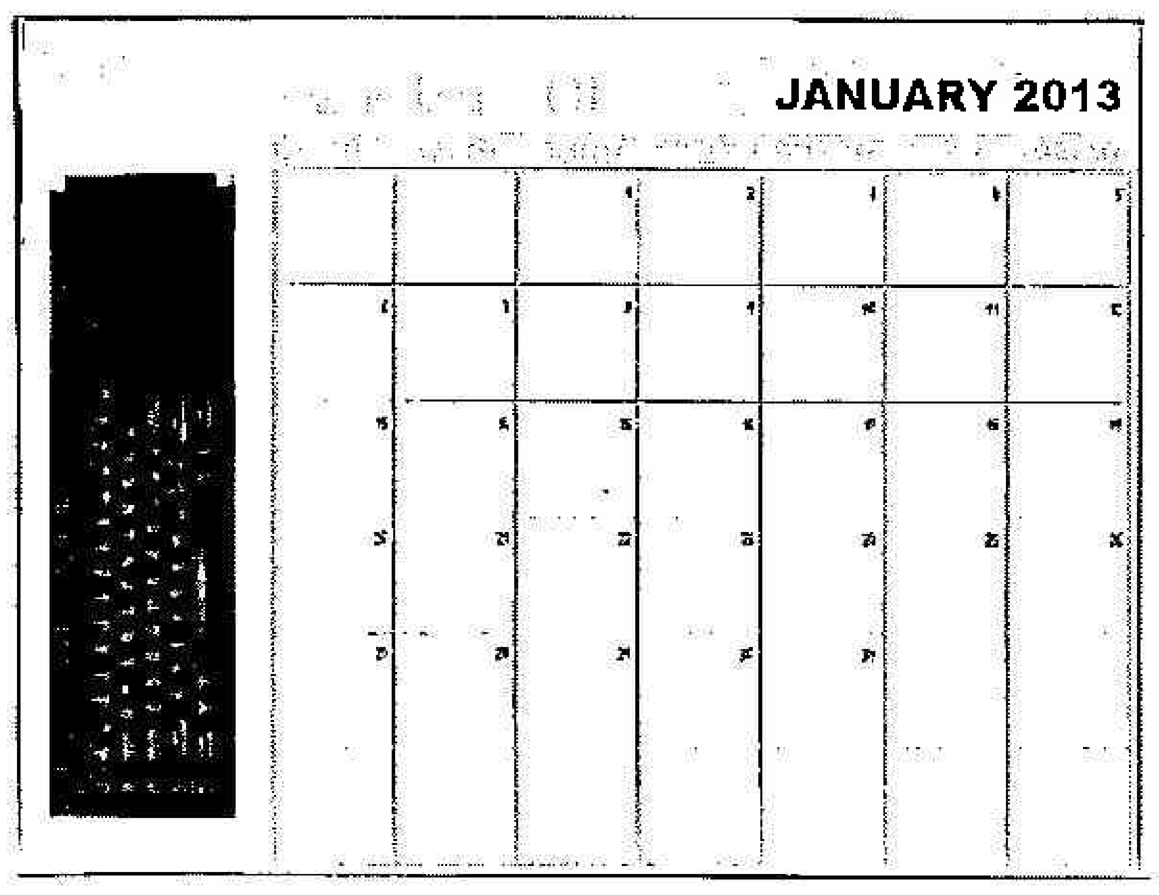
Gianni Carugno has released a project to build a mouse system for the QL. The 'GC' in the project name refers to his own initials rather than Gold Card. The set of PDF files contain circuit diagram, parts list, PCB layout, adapter lead details, firmware etc for building this project, which interfaces a PS/2 style mouse to a QL CTL port to emulate cursor key presses, like a QL joystick. This project is completely free to download. The author says that although a USB style PS/2 mouse can be used (usually one supplied with a PS/2 to USB adaptor) a purely USB mouse may not work. Gianni says that power may be taken from the QL EPROM slot (see the picture below) or via a low power PSU if you prefer.
Download it from the Hardware Docs page at:
http://www.dilwyn.me.uk/docs/hardware/index.html
[ED: A review of this project appeared in QL Today v17 i2 p6]



QaLendar 2013

My annual QL-themed calendar, The QaLendar, is now available for 2013.
It may be downloaded free as a PDF or Word 2010 file from
http://www.dilwyn.me.uk/gen/calendar/calendar.html





ITALIAN PSIONS

To add to the British, American and German versions of QL Quill, Archive, Abacus and Archive already available on the Dilwyn Jones QL website, there are now Italian versions available too. Download version 2.23 of them from:
http://www.dilwyn.me.uk/psions/index.html

BLOCKWALL GAME

Gianni has also been busy writing a Breakout-style game called BlockWall. He says it is his first QL game written in the Pascal language. The game runs on QemuLator but it is best to set the emulator to run at QL speed if it is used on a fast computer.
BlockWall is available to download from the Games page at:
http://www.dilwyn.me.uk/games/index.html

Sjef van de Molengraaf (secretary of the Dutch QL group Sin_QL_Air) has sent me an alternative version, available from the same page.

More calendar utilities (including SuperBasic programs to help create your own calendar) are available from the Utilities page on the website
http://www.dilwyn.me.uk/utils/index.html.

May

MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

2013

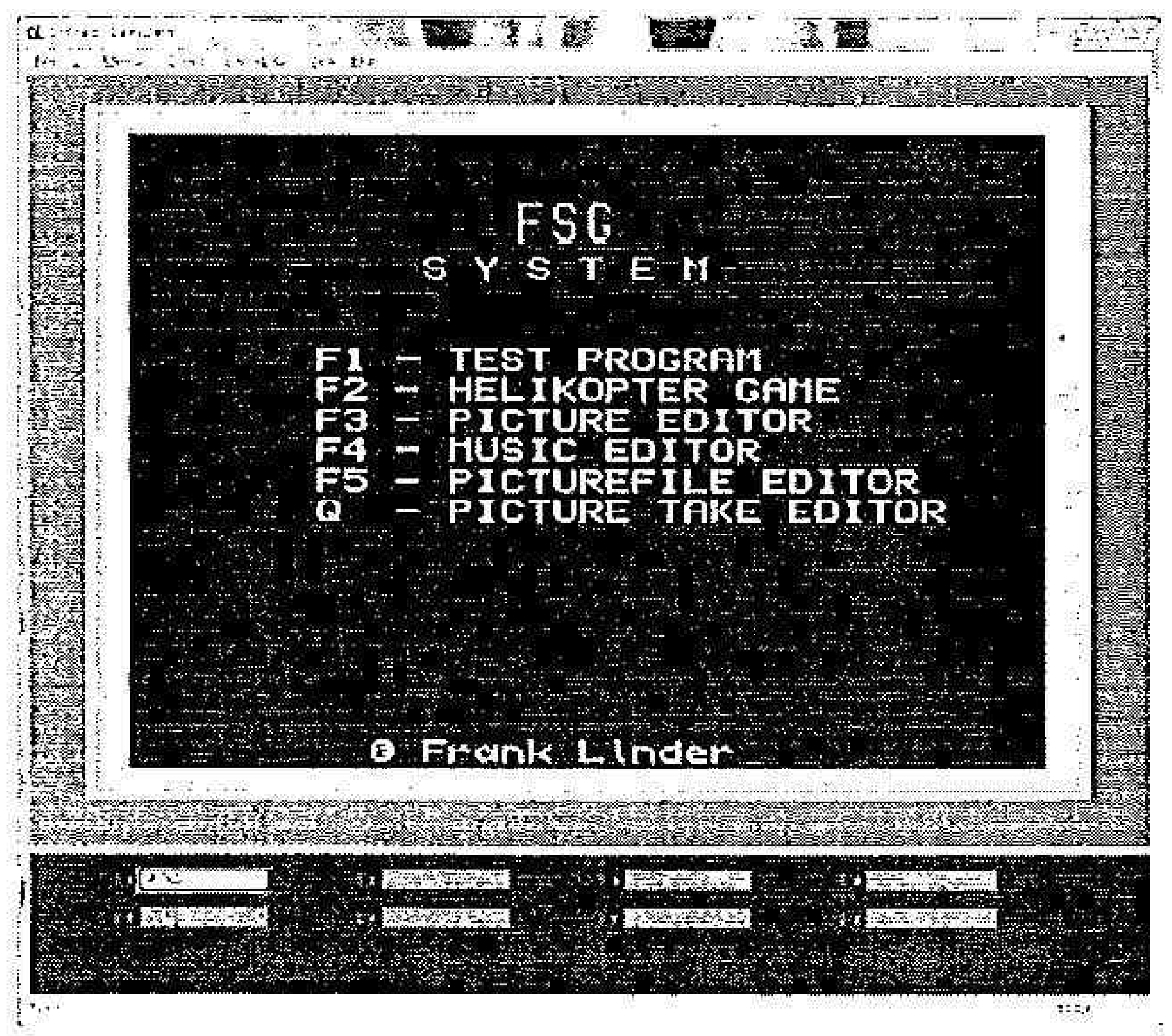
FRANK'S SPRITE GENERATOR

FSG is a sprites package for QL SuperBASIC, written by Frank Linder in Sweden. FSG stands for 'Frank's Sprite Generator'! It was written in the 1980s but never released commercially. Frank has now given permission for it to be released as Freeware.

The package includes an icon editor, screens editor and music editor, along with some very playable sample demo games including Heli (a helicopter game), Aste (based on Asteroids), Grott (a cave/mountain adventure game) and Combat (2 player helicopter game). Note that the package only runs on a system with QL mode 8 screen resolution and colour mode, with system variables at the address of an original QL system. Sprite control is via a comprehensive set of BASIC extensions, creating games which can run interpreted in SuperBASIC and compilable with Q-Liberator.

The zipped package is a 120KB free download from the Games page on my website at

http://www.dilwyn.me.uk/games/index.html



```
FSG
S Y S T E M

F1 - TEST PROGRAM
F2 - HELIKOPTER GAME
F3 - PICTURE EDITOR
F4 - MUSIC EDITOR
F5 - PICTUREFILE EDITOR
Q  - PICTURE TAKE EDITOR

© Frank Linder
```

QL ADVERTISING FONT

If you'd like a font which resembles the font used in original Sinclair advertising - the well known large letters 'QL' for example - I have now added one to my website. It may come in useful for QL devotees wishing to recreate historical articles about the QL, for example. Following a request made on QL Forum, Anthony Prime said it is a font called Saxony-Serial-Regular.

Download Windows True Type font, Type 1 and Line Design versions from the Fonts page on my website at

http://www.dilwyn.me.uk/fonts/index.html

On the same page you can also get a font designed by Lee Privett which will let you reproduce the QL screen font in a word processor document, along with the square characters used for the Sinclair logo - these fonts may come in useful if you wish to reproduce historical QL and Sinclair documents.

Sinclair QL

sinclair

sinclair QL

QUANTA NEWS - Tandata Modem Software

Many years ago, Quanta was able to supply the old Tandata three-stack modem systems for the QL. These were available with enhanced software, which included a cut-down version of the Qualsoft terminal programs from TF Services. Thanks to *Rich Mellor* who was able to locate a copy, and with thanks to *Tony Firshman* for confirming we can still supply it to members, this software can again be made available for those who wish to try out these classic modem systems on a QL, as they do become available on eBay and SellMyRetro from time to time. The software is not part of the Quanta library, rather it will be held by Treasurer *John Gilpin* should any member wish to obtain a copy. The software is free of charge to members.

Other Software

GWASL

George Gwilt writes:

'Norman Dunbar has found a fault in GWASL which affects the reporting of errors in labels. This is corrected in version 2.05 which can be downloaded from my site at

http://gwiltprogs.info/

SMSQ/E MOVE

Wolfgang Lenerz writes:

'SMSQ/E v. 3.14, with the new licence, has been moved to:

www.wlenerz.com/smsqe

There isn't really much that is new in 3.14, except that it contains the SMSQ/E code for SMSQmulator - no need to upgrade except for that.'

From the Distant Past

QL Today's editor has received an email from *Stephen Denson*, a former trader from the earliest days of the QL. He sold business software including QL Invoicer. Stephen has recently bought some back copies of QL Today on eBay. He comments:

'I am amazed at the professional quality of QL Today magazine which serves what must now be a fairly small community which is seemingly split between original black box users and those running advanced emulators and o/s on modern PCs. Reading between the lines in your 2007 issues (which is as far as I've got) the market was shrinking rapidly then yet both QL Today and Quanta are still going.'

BILL RICHARDSON

Bill Richardson, one of the most respected and liked former QL Traders, has died aged 94.

Bill had an association with the QL from the earliest days and continued trading until 2006. In a 2002 video interview with Darren Branagh he tells of his link to Clive Sinclair starting from the latter's pre-computer days. Both were involved in digital watches, and at one stage Bill was a rival trader.

Bill's involvement with Sinclair products started with the Spectrum. He estimates that there was a return rate of between 8% and 15% and Clive Sinclair had made no provision for their repair. Bill took over the responsibility as an intermediary and arranged for the returns to be repaired and resold.

Initially he was involved in the QL in a small way buying and selling about 50 at a time. However he seized his opportunity when a large QL order destined for Iran collapsed and he was able to buy 4,000 cheaply. In later years he mainly traded in ZX88's.

The video interview can be viewed on YouTube:
http://youtu.be/P3CwlineYZ4

Many tributes were paid to Bill on the QL users email list. In announcing Bill's death Tony Firshman reported:

"He and Felix Fonteyn (brother of Dame Margot) were responsible for many thousands of effectively new QLs, made from ex-Sinclair stock. He negotiated a batch of new QL membranes from the original manufacturer, and I split the cost with him. I think it was 5,000 in total. I also did a lot of Z88 work for him. He would have been the first to admit he was not too technical, but was a miracle worker in locating products and getting people with the know-how to work for him.

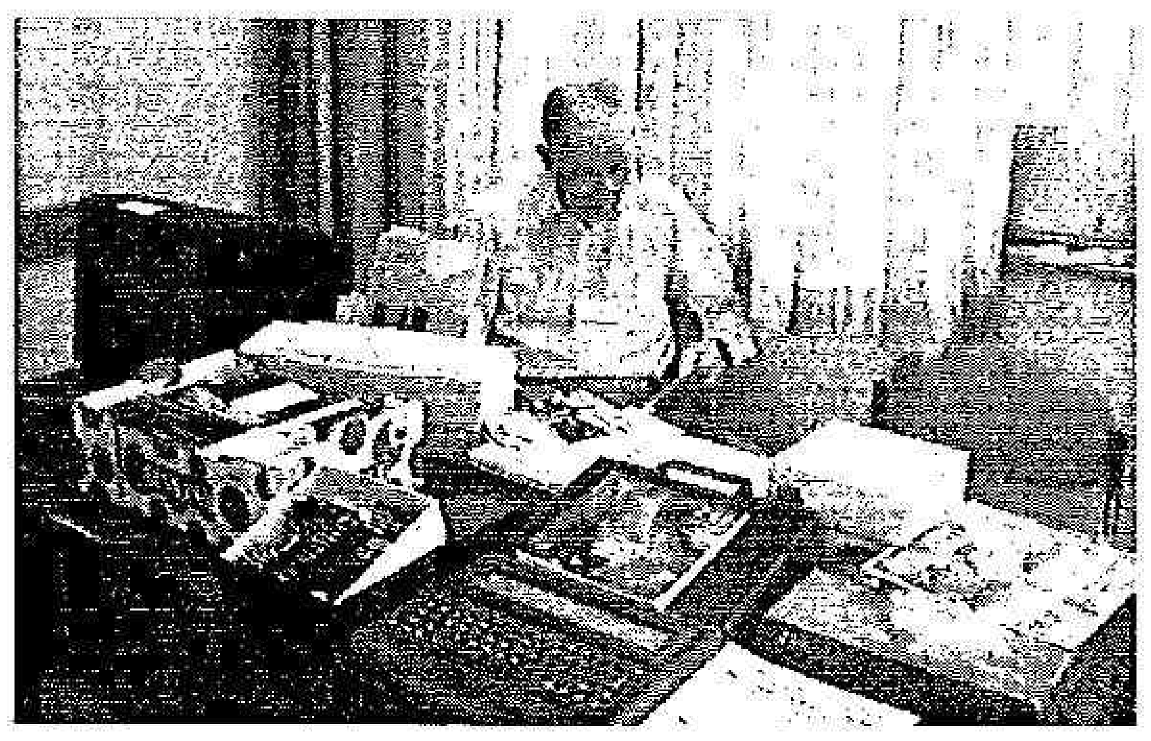
I saw him many times in recent years. He was still active and walking (very slowly) and with a stick, when he could be persuaded to carry one. He was though quite frail. His mind was still OK, but again not high speed!"

Tony has posted a gallery of recent photos of Bill on his website:
http://tfs.firshman.co.uk/temp/bill/

John Taylor commented:

"I first met Bill at the ZX Fairs in the Horticultural Hall. His stand was always a profusion of software, hardware and little bits and pieces, manna from heaven to the eager convert. In many respects Bill stood head and shoulders above the rest of us."

Rich Mellor described him as a "Real Gent" adding that it was always a pleasure to speak with him and John Southern as "a perfect gentleman who had time for everyone".



Bill's last QL show was to be Quanta's QLis21 in 2005, although he attended a show in Portslade a year later. When I asked Bill if he wanted to be included in the QLis21 flyer, he emailed me:

"The QL may be 21, but I am 87 and it's time I should pack up, having sold over 4,000 QL, and 10,000 Spectrums and associated bits and pieces. I have had a great time, made many friends and I am now spending more time on the fringes of politics and writing, so very sincere thanks to all the friends I have been privileged to meet over 30 years or so."

Bill Richardson's wife, Dorothy, predeceased him in 2005. He leaves a son and a daughter.

Bill Richardson - a "real gentleman"

by Tony Firshman

William Newall Richardson sadly died on January 3rd, 2013 aged 94.

Most of us will have known him or purchased from the company he ran after his **fourth** 'retirement' in the mid 80s.

Hê did eventually retire and lived his many remaining years in a magnificent sheltered apartment in Luton. .I visited him there for lunch very many times with **Vic Gerhardi** (Z88 trader). .We even had a day out to Southend in 2011 with Vic, his friend Denise and his father. .They all have some degree of physical disability and used sticks. .Bill also was frail, walked very slowly, and with a stick if he could be persuaded to use it. .We had some hilarious **slow** convoys in and out of establishments. We all crammed into the smallest of the many Southend Rossi establishments, blocking it completely and ate ice creams, messily. .The pier directly opposite was closed as a barge had hit it the previous night. Sadly our 2013 trip was to take Bill to the end of the pier.

Bill was born on April 18, 1918 in East Riggs on the Solway Firth. .As his daughter Sue puts it, slightly tongue in cheek: 'This of course makes him Scottish'. .His family though soon moved to Swinton in Lancashire, where he went to school. .I did not know this, and it explains the lack of a Scottish accent! .His retirement project was to write a book on the history of the Scots. .Sadly, when I offered to proof read, I saw it had not got beyond the introduction page. Many will remember the tartan cloth covering his stall at QL shows.

He excelled in mathematics at school and studied Electrical and Mechanical Engineering at *Salford Technical College*, now (of course) part of the University of Salford (Peel building). It is a magnificent Victorian edifice reminiscent of the Natural History Museum in London.

He spent the second world war in Gosport and Southend doing engineering work for the Royal Navy.

He married Dorothy in 1947, and his daughter Sue was born in 1948. .They moved to 6 Ravensmead, Chalfont St Peter, Bucks in 1952 and Nick was born five years later.

I remember 6 Ravensmead and Dorothy very well. I visited his home many times to share a car to QL shows, and to discuss QL and Z88 projects. After Dorothy died in 2004, I also did quite a bit of work on his house and garden. .He bought a plot of land in the 50s, built the road and his magnificent large detached house at the end. .I believe he was also an instrumental part of the housing development there.

He had a long and varied business career. He retired three times, according to his daughter, because he always got bored with gardening and golf, and Dorothy got fed up with him being at home!

Bill worked for a number of international companies including Plessey, GEC and Hughes Aircraft Company. .He started the first of his own semi-conductor distribution business' which he sold in 1969 and 'retired' for the first time. His last job as an employee before his second 'retirement' was marketing director of Chubb. Shortly after that he re-emerged as the managing director of an international legal publishing company. Following his third 'retirement', he started his own software and book publishing company called **Oxford Computer Publishing** (OCP) from an old house in *Dell's Secondhand Car Lot* in Chalfont St Peter. I saw those premises, and it was not quite the 'Arthur Daly' place his daughter likes to call it! .His fourth 'retirement' came when he closed OCP..However it resurrected as *EEC* and *W N Richardson & Co* and it is this that most of us will remember him for.

Many will remember his son Nick accompanying him to QL shows, when he needed help with the sheer weight of his product.

He did remarkable work for the QL and Z88 scene from the mid 80s. He had many contacts with *Sinclair Research* and *Cambridge Computing* (Z88) when they were active. When both companies ceased trading he procured a great deal of their stock. This allowed his colleague **Felix Fonteyn** (brother of Dame Margot) and others to assemble effectively new QLs in original packaging. He sold thousands of these. He also arranged for production of 5000 QL membranes from the original manufacturer. Such was his generosity, he split the batch with me and charged me cost only. I also made a lot of product for him for the Z88 and QL. One memorable job was to make 100 microdrive extension leads. These were with ribbon cable that could be twisted so that Spectrum drives faced forward. It was immensely hard sourcing the necessary edge connectors, which had to be custom made in the USA. Unfortunately he mislaid the finished product. I remember searching a number of times in the piles of stock around his house. We never found them. I did though find plenty of things even he did not realise he had, like a huge pile of 8301 chips - much in demand then, and scarce. I often wonder whether the microdrive leads emerged when he sold his home.

One thing I will always be grateful to Bill for is the introduction to **Vic Gerhardi** (Rakewell Ltd) who trades with the Z88. Vic was considering a flash memory card for the Z88 and Bill suggested me as the pcb designer and manufacturer. That was in the early 90s, and Vic has since become not only a business colleague, but a very good friend. I showed the flash card for the Z88 to **Stuart Honeyball** (Miracle Systems) at an Eindhoven show. We travelled back together, and by the time we reached England we had mapped out RomDisq. So indirectly Bill was even responsible for RomDisq! I don't think I ever told him that.

In his latter years, after Dorothy died, I am told Bill took to going on cruises. These were regarded with great trepidation by his family, as he was always joking about getting married again. His son Nick accompanied him on these adventures. This was very brave of him, I am told, as Bill's last travelling companion had been "mislaid" in the Arctic Circle. Nick would hold him in check, vet any lady friends, and smooth over the chaos that seemed to follow him everywhere. This came as a great surprise to me as I had never seen this side of his character. From this and other things his family have said, he sounded a great deal of fun. I wish I had seen more of that side of him.

He was very lucky to live a very full, active and healthy long life. He will be sorely missed.

Changes...

by Jochen Merz

Where do I start after the obituaries? I feel I need to add something as well.

I knew Bill for many many years ... and yes, he was a real Gentleman. One could rely on his word. Although I "only" met him (many, many) times at the various QL shows, I remember many interesting conversations with him - at the shows or after the shows.

One thing I remember very well was the way he used to talk about the UK keeping the Pound. I agreed and I thought in a similar way. I even bet on the idea that the UK will not give up its currency, like we in Germany were forced to.

I remember that he gave me flyers and stickers, saying "Keep the pound - save democracy." How right he was. We all seem to lose democracy, but I feel that history books will name the EURO to be the worst mistake.

Although Bill was more into QL hardware and even more into Z88 products, it was always great talking to him. Yes, he will be missed.

... a short break...

Next the answer to the question which was printed on the front cover of issue 2. The "When?" about the Hamburg QL meeting. That's something I don't really know. Not this year, that's for sure now.

When I initially asked the newsgroup in November about a date for the planned meeting in Hamburg, I got some replies. Not many, but some. Then, QL Today was shipped in December and I got exactly ONE additional reply ... that's all. I was hoping for a few more replies, but until today, end of February, that was all I got. Disappointing, but reality.

I spoke to Rainer Wolkwitz, who offered the rooms for the venue, and we both thought we might end up sitting there together, hoping for one or two visitors to arrive.

Hamburg is always worth a visit, but my wife and I feel it is not feasible to drive to Hamburg, stay two nights, spend the day in between in the room, hoping for a visitor to arrive, and drive home. Not the way we currently feel, health-wise. We have cut down considerably on travelling ... we both feel too tired and exhausted.

Rainer said that the rooms will be available in 2014, 2015 and probably the following 4 years or so. Which means, a meeting in Hamburg may not necessarily be completely cancelled, but moved forward to next year ... who knows.

Maybe it needs more time than 5 or 6 months in advance. Maybe travelling is something we all try to avoid - maybe we all are getting older and travelling is just too expensive nowadays. Maybe there is no interest in getting together, as there is not much news nowadays. Feedback is most welcome!

The next change is related to the next issue of QL Today. We explained in the last issue that the postage in Germany roughly tripled as of January this year, because Deutsche Post do not offer the services I used to use anymore. This means, that QL Today cannot be offered at the current price. As it would be a considerable increase, Geoff and myself discussed the situation and we both thought, time has come to decide that QL Today in its current form will end.

I remember that many years ago, I spoke to Roy Wood and we thought: well, if the readerships falls under 400, we stop. It did, but we continued (OK, easy for him to say that as I did the main work). Same for 300. We continued. Same with Geoff ... the readership is under 200 for some years and we still continued.

However, there are things we cannot change: the costs. Printing costs have gone up, shipping, so the price-per-issue is too high now.

We would like to end the volume (and QL Today) in a way that the last issue will come in paper and electronic form. Producing CDs/DVDs takes a long time ... and as the summer time has always been quiet in the past, Geoff and I thought that we will skip summer this time and deliver the final issue 4 of QL Today in September. Both on paper and on digital media.

We have stretched the deadline slightly ... but remember, that once the paper issue is done, it needs to be converted and then the CD has to be produced.

If you feel you would like to send us a comment, letter, article, to be printed in the next and last issue, please do so. I don't mind getting over the 32 pages-limit again - as far as I am aware, we did this in every issue since it had been announced ... well, better for the readers than having a much thinner issue, which would have led to a much earlier end of QL Today.

If you now ask, if this really is the end of QL Today, then my answer will be a definitive "Yes". Do I want to see somebody to continue? Not really, to be honest. If somebody wants to start something "fresh" - up to him. During these 17 years, I have tried very very hard to be as reliable as possible and deliver what was paid for. I relied on the help of others, all the authors, of course, Stuart, Roy, Dilwyn, Geoff, Bruce etc. - and we all did very well. I do not want to see QL Today connected to something negative, ending like "IQLR", for example, so let's end it.

I am feeling rather sad while I write these lines - QL Today has been a major part of my life... after all, I produced it for 17 years and especially in the early years, we had six Issues per year - both English and German! How did I manage it these days, time-wise? I don't know.

Enough changes and bad news, you will think. Yes, I think so too. That's enough. My website will remain, the products will still be offered (for at least as long as my old EPSON laserprinter will survice). Maybe I will find a bit more time after QL Today to convert documents from text87 EPSON GQ mode to PDF, update some products, fix some bugs. Would be great. I do miss programming in assembler. Will probably take some time to get into it, but it has always been a pleasure. So whatever happens it will hopefully be reported in the Quanta magazine, on my website **SMSQ.J-M-S.com** and I will report news, updates and changes to the QL Newsgroup, of course.

JOCHEN MERZ SOFTWARE

Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Phone +49 203 502013
Fax +49 203 502012
EMail: SMSQ@J-M-S.com

Web: SMSQ.J-M-S.com

QPC2 Version 3 + SMSQ/E Software QL-Emulator for PC's EUR 59,90

QPC2 Version 3 - Upgrade from QPC2 Version 2		**EUR 19,90**
QPC2 Version 3 - Upgrade from QPC2 Version 1		**EUR 39,90**
QPC Print - printer emulation driver for QPC		**EUR 39,90**
BUNDLE: QPC2 and QPCPrint	**ONLY**	**EUR 79,90**
Agenda Agenda program for WMAN and Prowess	[V1.09]	**EUR 14,90**
Suqcess Database front-end for WMAN	[V2.05]	**EUR 19,90**
QD2003 Pointer-Environment-Editor	[VB.01]	**EUR 29,90**
QD2003 Upgrade from Version 9 and older	[VB.01]	**EUR 14,90**
QMAKE Pointer-driven MAKE for GST/Quanta Assembler	[V4.31]	**EUR 14,90**
BASIC Linker	[V1.21]	**EUR 14,90**
WINED Floppy/Harddisk Sector- & File-Editor	[V1.26]	**EUR 14,90**
FiFi II File-Finder - Extremely useful!	[V4.31]	**EUR 14,90**
FiFi II Upgrade from Fifi Version 3 or older	[V4.31]	**EUR 9,90**
EPROM Manager	[V3.02]	**EUR 14,90**
QSpread2003 Spreadsheet Program	[V4.04]	**EUR 29,90**
QSpread2003 Upgrade from Version 3 and older	[V4.04]	**EUR 14,90**
QPAC I Utility programs	[V1.11]	**EUR 19,90**
QPAC II Files, Jobs & other Things	[V1.45]	**EUR 29,90**
QTYP II Spell checker	[V2.17]	**EUR 19,90**
QPTR Pointer Toolkit	[V0.30]	**EUR 29,90**
DISA Interactive Disassembler	[V3.04]	**EUR 29,90**
CueShell	[V2.14]	**EUR 29,90**
CueShell for QPC	[V2.14]	**EUR 14,90**
SER Mouse software mouse driver for serial mice		**EUR 10,00**
EasyPTR Version 4	[V4]	**EUR 59,90**
EasyPTR Version 4 - Upgrade from earlier versions	[V4]	**EUR 39,90**
QDT - QL Desktop program		**EUR 59,90**
QMENU Version 8 - with new, printed Manual	[V8.02]	**EUR 24,90**
QMENU Version 8 - Update from earlier Versions, also with printed manual		**EUR 17,90**
QMENU Version 8 - New/Update for QL Today subscribers, with prtd manual	**ONLY**	**EUR 14,90**

Please add EUR 4,90 for postage to all destinations - Germany, Europe, Worldwide!

We accept VISA, MasterCard & Diners Club online and offline!
Details for money transfers:

- Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43
- Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000
- Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000
- The Netherlands: Jochen Merz, Gironummer 3258439, Postbank NL Amsterdam
- and from all other countries in EUR with IBAN and BIC to account
  Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360
- UK customers can pay in £ (convert EUR prices above to £ by multiplying with 0.83) to
  Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
  or send cheques in £ - no fee for UK sterling cheques!
- If you wish to pay via paypal, send money to Paypal@J-M-S.com

Cheques payable to Jochen Merz only!

Price list valid until 31st of May 2013

SMSQmulator - THE FIRST STEPS

by Geoff Wicks

I have a bone to pick with Wolfgang Lenerz. Last May he announced he was writing an SMSQ/E emulator for Java. He released SMSQmulator towards the end of January on the day before I was due to be away from home for a couple of days. I arrived back, weary from two nights sleeping on a boat, and having coped with severe public transport disruption through ice and snow in two lands, to discover no fewer than 85 emails in my inbox. Almost all had the subject line "SMSQmulator". Clearly Wolfgang had aroused the QL community from its winter hibernation.

Until now there has been very little interest in Java among the QL-ers. However it is a programming language used in many devices and an emulator for Java raises the possibility of a QL platform for mobile devices.

You will almost certainly have Java installed on your PC, but, if not, you can download a copy from:

www.java.com

The emulator itself can be downloaded from:

www.wlenerz.com/SMSQmulator

(Wolfgang warns that this address is case dependent)

There are two versions of the emulator, one for Java 6 and the other for Java 7. Wolfgang writes that "both versions are functionally identical. However, I have noticed that under Java 6, some Linux machines will NOT play sounds, where they will under Java 7".

If you are uncertain which Java version is on your machine you can always upgrade to Java 7 from the Java website. It was quicker to do this on my laptop than investigate which version of Java was already installed.

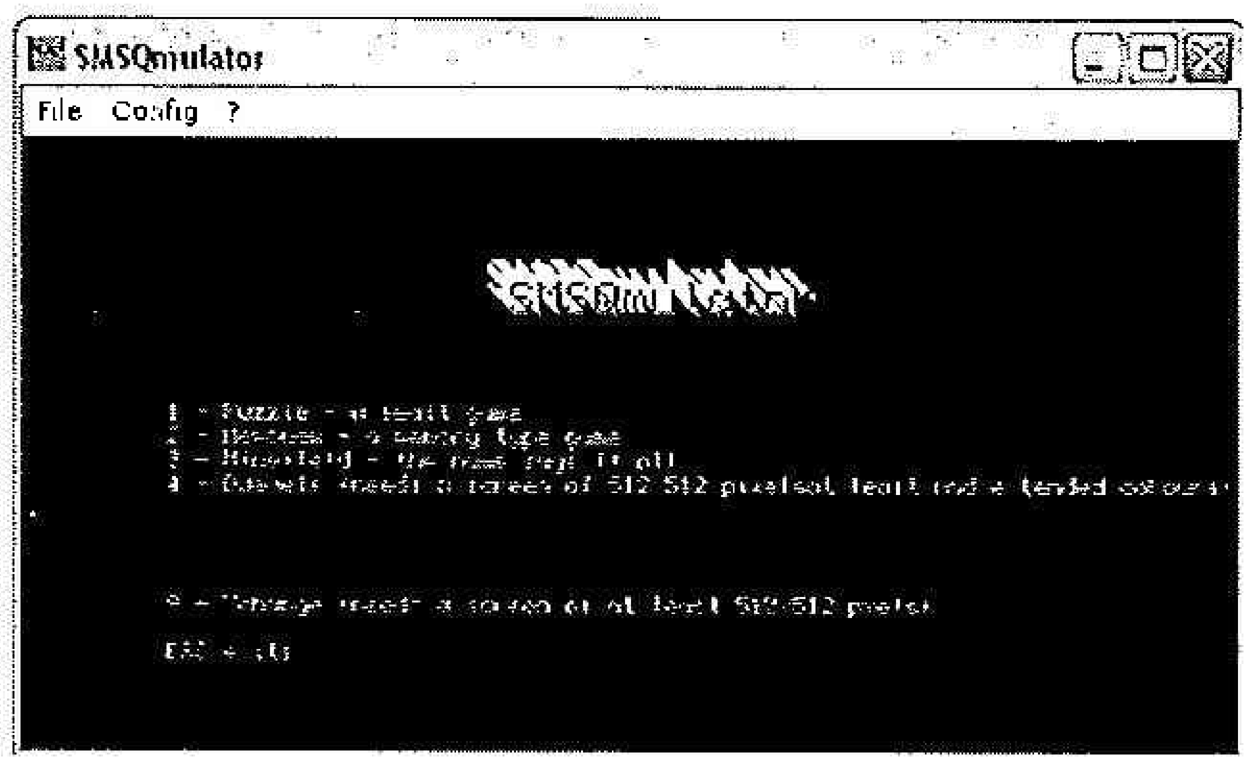
The emulator comes in a 1.56 Mb zipped file containing the emulator itself, SMSQ/E, a WIN file, a subdirectory called "lib" and two manuals, one by Wolfgang and the other by Tim Swenson, who did much of the beta testing. Before unzipping the file it is useful to have a folder called SMSQmulator.

Wolfgang's manual is 31 pages long and Tim's 23 pages. Both are pdf files and both are legible on my e-reader. Wolfgang starts his manual with the words:

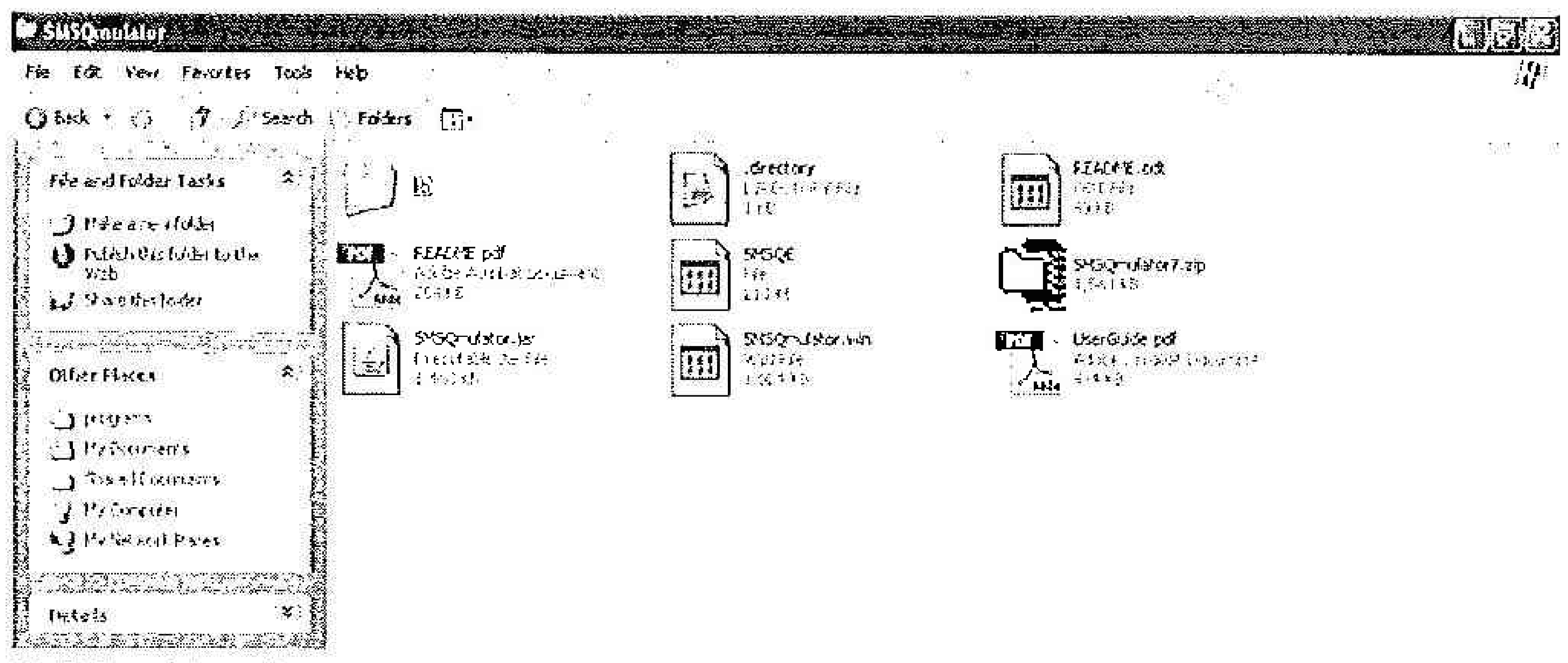
"Please read this ENTIRELY before you do anything else."

I wonder how many people did that. He does warn, however, that sections I to VI are essential, and it would be difficult to use the program without referring to them.

Loading SMSQmulator is usually very simple and is just a matter of clicking on the SMSQmulator.jar file. It usually loads quickly and, as supplied, after a brief sighting of the familiar white and red screens, loads a menu for 3 QL games, 1 SMSQ/E game and Xchange. The default screen is the standard QL 512 x 256.



Wolfgang's default screen

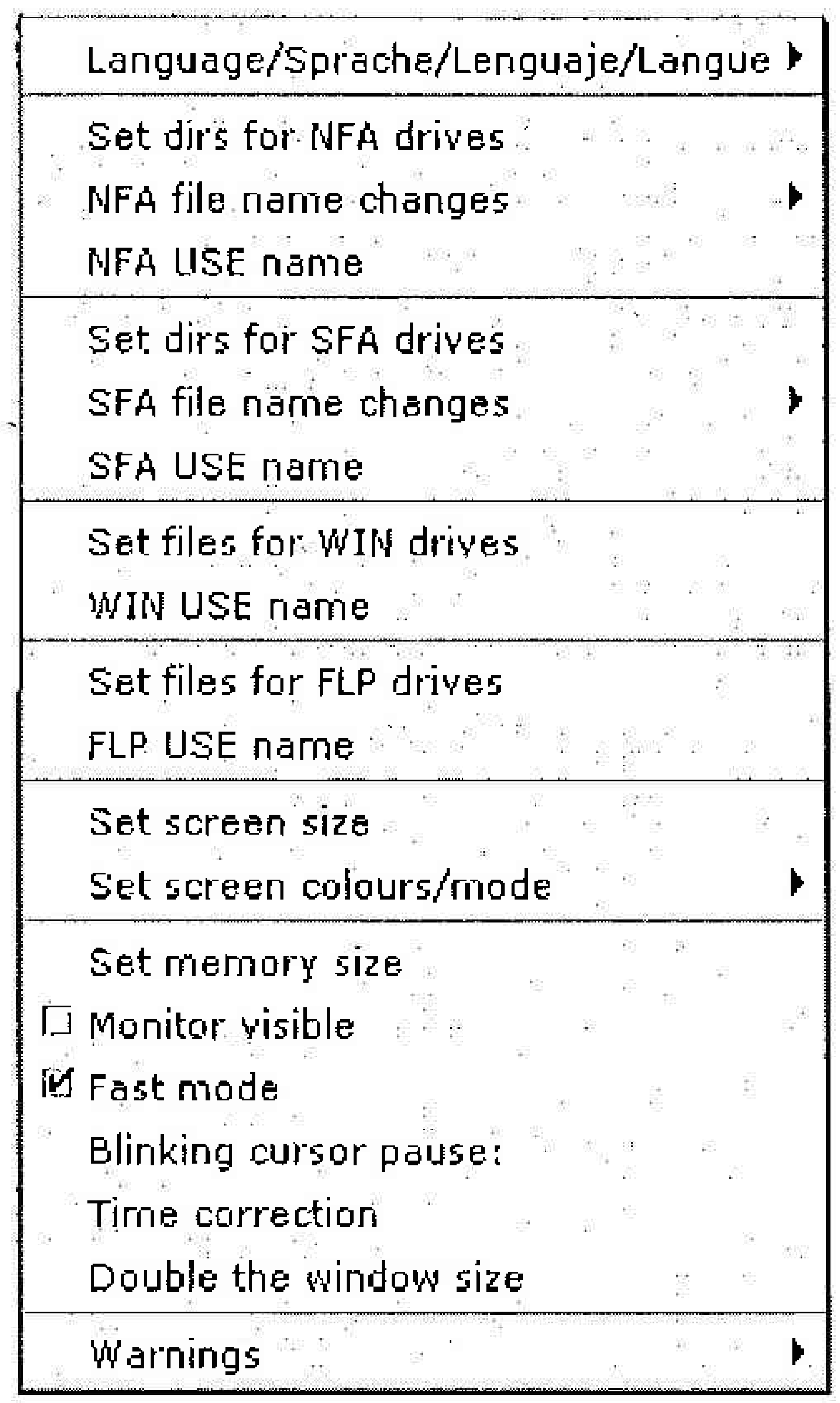


SMSQmulator directory

It is worthwhile to spend some time exploring this screen to familiarise yourself with this program before moving onto something more ambitious.

Initially you should look at the first three menu items only, because the other two require configuration changes. The three items are 'old fashioned' QL games.

When you have finished playing with these it is time to move onto Xchange, which is the Marcel Kilgus version using a 512 x 512 screen. Click on 'Config' at the top of the screen and a pop-up menu appears.



Configuration Menu

Most of the items on this menu will be meaningless as we still have a lot to learn about the program. You should click on 'set screen size' and change this to 512 x 512. Any configuration changes you make are saved and will be the configuration the next time you load the program. On occasions you may have to click reset or even exit the program and restart it to make the configuration changes operative.

You can now load and experiment with Xchange. Try, for example, some simple word processing or a simple Easel graphic. You cannot load or save files at this stage because we still have to learn how SMSQmulator handles devices.

The next stage is to look at the QJewels game. So far all the programs we have looked at use original QL colours. QJewels uses high colour mode and we have to reconfigure SMSQmulator to play it. Go into the configuration menu and amend 'Set screen colours/mode' to 16 bit colours. To make the high colour mode active you have to end SMSQmulator and restart it afresh. Once you have made the change to high colour it is worthwhile to keep SMSQmulator in that mode.

When you have finished looking at Wolfgang's menu program it is time to start planning your own program use. Before you can do this you must first master SMSQmulator's device handling.. There are four different device types, WIN; NFA (Native File Access); SFA (SMSQ/E File Access); and FLP.

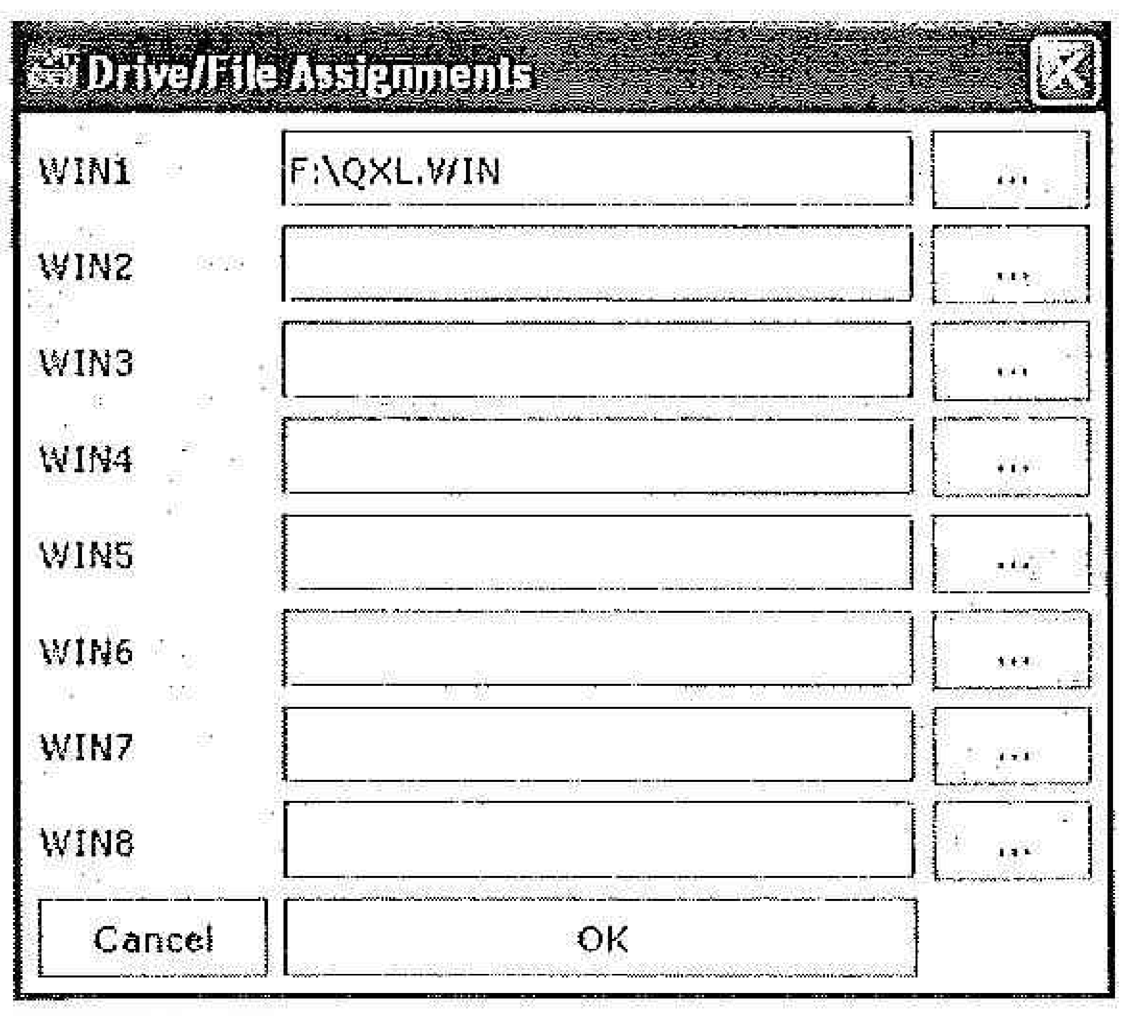
If you are unfamiliar with this terminology, don't worry, because if you are a QPC2 or other SMSQ/E user you will already be familiar with the concepts. In QPC2 terms examples of WIN would be win1_ and win2_ ; of NFA dos1_ and dos2_ ; and SFA flp1_ and flp2_.

The difference between an NFA and an SFA device is that NFA devices cannot save all QL files correctly. For example, you can save text, basic and compressed files, but not _exe and similar files.

It was probably a little unwise for me to make a comparison between QPC2 and SMSQmulator because there are large differences over the handling of SFA devices.

This is best explained by my own experience when testing devices on SMSQmulator.

For the WIN device I used the same QXL.WIN that I use for QPC2. SMSQmulator booted successfully from this device and most of the programs on it ran under SMSQmulator.



WIN1: F:\QXL.WIN

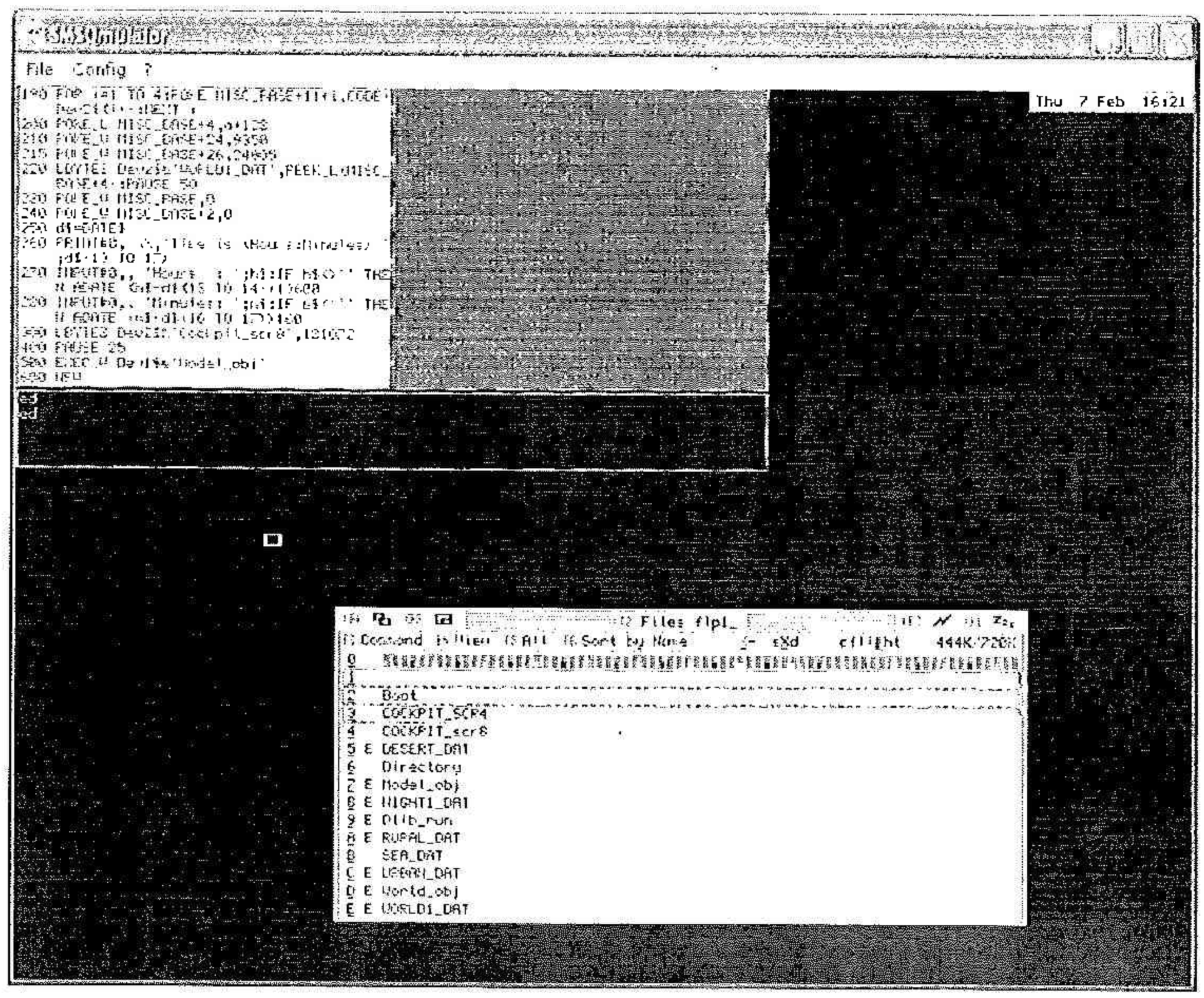
WIN Drive Menu

For the NFA device I used a USB memory stick almost permanently attached to my PC. I use it in a similar way that I used to use floppy disks, that is for temporary and back up storage of a wide variety of files. QPC2 and SMSQmulator both access this device in the same way. The device has a small number of SuperBasic programs on it and I was able to run them directly from this device.

SMSQmulator SFA devices are, however, different from QPC2 SFA devices. If I put a floppy disk in my PC, QPC2 would be able to read and write to it whether it was a QL or a PC formatted disk, although some QL files would be corrupted on the PC formatted disk. SMSQmulator would read and write to a PC formatted disk, but would not recognise a QL formatted disk at all. In SMSQmulator SFA devices are not the physical device as such, but the way in which files are stored on that device.

I set up my SFA device on a small memory card fitted into a card slot on one of my printers. I copied a file, postcodes_exe, from win1_ to the device, sfa1_. I could then run postcodes_exe directly from the memory card. If I tried to run postcodes_exe directly from the memory card on QPC2 it would not work, because QPC2 would not recognise the file type. Equally if I had transferred postcodes_exe from QPC2 it would not run in either QPC2 or SMSQmulator.

The biggest difference between QPC2 and SMSQmulator is in the use of flp. QPC2 looks at the physical disk, but SMSQmulator does not. The latter has to look at an image of a disk and that image does not have to be on a floppy. There are programs, such as Rawread, that allow you to make an image of a disk on a PC device.



```
190 FOR i=1 TO 4:POKE MISC_BASE+11+i,CODE(
    Dev$(i)):NEXT i
200 POKE_L MISC_BASE+4,a+128
210 POKE_W MISC_BASE+24,9358
215 POKE_W MISC_BASE+26,24905
220 LBYTES Dev2$&'BUFLDI_DAT',PEEK_L(MISC_
    BASE+4):PAUSE 50
230 POKE_W MISC_BASE,0
240 POKE_W MISC_BASE+2,0
250 d$=DATE$
260 PRINT#0,\\,'Time is (Hours:Minutes) '
    ;d$(13 TO 17)
270 INPUT#0,,'Hours : ';h$:IF h$<>'' THE
    N ADATE (h$-d$(13 TO 14))*3600
280 INPUT#0,,'Minutes: ';m$:IF m$<>'' THE
    N ADATE (m$-d$(16 TO 17))*60
390 LBYTES Dev2$&'Cockpit_scr8',131072
400 PAUSE 25
500 EXEC_W Dev1$&'Model_obj'
590 NEW
```

```
Files flp1_
Command in Files  All  Sort by Name   444K/720K
0 ...
1
2   Boot
3   COCKPIT_SCR4
4   COCKPIT_scr8
5 E DESERT_DAT
6   Directory
7 E Model_obj
8 E NIGHT1_DAT
9 E Olib_run
A E RURAL_DAT
B   SEA_DAT
C E URBAN_DAT
D E World_obj
E E WORLD1_DAT
```

Directory and boot program from flp image

As it happened I had an flp image on my working USB memory stick. By pointing SMSQmulator to that, I could view the directory and load the boot file. (In this case I would not have been able to run the program as it is an old game only runnable on native QL hardware.)

Please note that you cannot write to an SMSQmulator flp device. You can only read from it.

I would strongly advise you to carefully read the sections of the manual dealing with devices - III, IV and V - because there are a lot of nuances and advice on their use.

If we return to the configuration menu, a lot of the items that were meaningless when we first looked at it, now begin to make sense. We can also see how to attach a device to SMSQmulator. All you do is to click on the menu item of the type of device and then a menu comes up showing the list of devices. You can, if you wish, click on the dots on the right of the device number to navigate through to the directory, but I found this a slow process. It was quicker and easier just to type the directory in.

You now know enough to make your own use of SMSQmulator, but how compatible will it be with programs you use?

The first thing to stress is that SMSQmulator is not a QL emulator and is not intended to be a QL emulator. It is an SMSQ/E emulator and thus will not be compatible with programs, such as old games, that will not run on other SMSQ/E systems.

Other than that it is highly compatible. When I attached my working WIN device to SMSQmulator, it ran the boot program I use for QPC2 loading, among other things, QPAC2 and the QTYP extensions. Practically all programs on the WIN file ran without problems including, I am pleased to say, the entire Just Words! range.

Very few incompatibilities have been reported and the main problems have been with some graphics programs and ProWesS. However Wolfgang has quickly and efficiently followed up any problems and the very latest version now loads LineDesign. I am still having difficulties with the graphics programs, but this may be a problem with my system because others are not reporting the same problems.

Another question is the speed of SMSQmulator. The only thing that can be definitely said is that it is faster than the black box, but much slower than QPC2. Wolfgang suggests that SMSQmulator should be about one tenth of the speed of QPC2, but users have reported wide variations. To give you an example SMSQmulator is one tenth of the speed of QPC2 on my laptop, but only one seventeenth on my desk top.

The reason for this is a little uncertain. Some have suggested speed problems are inherent in Java, but others have said these problems were only characteristic of early Java versions. Another suggestion is that the specific hardware, single core or multicore processor, of a PC could have an effect. (My laptop is dual core, but my 10 year old desktop is single core.) SMSQmulator does seem to hog processor time and is probably easier to run on a multicore machine. Wolfgang says that Java only uses one core on a multicore processor, which means that it is unlikely to affect the performance of other programs.

Tim Swenson commented:

"On all systems that I have run SMSQmulator, it really sucks up the CPU time. There is an option to have it take up less CPU time when the cursor is blinking.
Even when running 100% CPU, I've found the other applications are not greatly impacted and are fairly responsive. I just noticed my fan kick in a lot more often."

In summary, SMSQmulator is a remarkable piece of software. Given its complexity the number of bugs reported has been very small. If nothing else Wolfgang Lenerz has given the QL community not just a new product by also a welcome boost to our morale.

More discussions with George & Norman

by George Gwilt and Norman Dunbar

Norman's response to George's Letter to QL Today re Norman Dunbar's Article on Assembler - Part 31, Part 3. Norman's *[ND]* answers to George's *[GG]* comments on Assembler - Part 32. Comments inserted by the Editor are marked *[ED]*.

[GG] In Part 3 of Part 31 Norman gives the code for the four utilities missing from his program so far. As usual I have a few, mainly minor, comments.

[ND] Minor? I like minor, it means I haven't screwed up too badly! ;-)

[GG] 1. In both the routines cp_string and ap_string on pages 25 and 26 which deal with strings there is a branch to the exit if the string length is zero. In fact that branch is not needed since in both cases the program will exit via the later dbf instruction and no copying will take place.

[ND] This I agree with. And yes, it does waste the odd couple of clock cycles, however, I do try to be explicit in my coding - both at work and at play. Hence the habit of checking for stuff I don't want and baling out early when it is found. However, it is a point to bear in mind when every byte is sacred. Like back in my 1Kb ZX-81 days!

[GG] 2a. In iw_print the instruction setting D3 to -1 after the label iwp_prnt is only needed if a branch to iwp_prnt is in fact made from anywhere. I cannot detect any such branch.

[ND] Yes, D3 need not be set at that point as it is preserved by the previous trap #3 to clear the window.

[GG] 2b. Also, I suggest that the testing of D0 should appear before, not after, iwp_exit. The reason is that if the branch to iwp_exit is taken then at that point D0 contains the non-zero item #io_sstrg and not the error return from the corresponding trap #3.

[ND] This is a bug in my program library. I shall write up a correction in the next article. Thanks George. My suggestion would be to move the instruction moveq #io_sstrg,d0 from its current location at iwp_prnt and place it after the instruction beq.s iwp_exit. That way, D0 is zero when a return from CLS is made and if there is nothing to print, we exit with D0 correctly set to zero and the flags set accordingly.

[GG] 3a. In iw_input I suggest that the four lines starting

```
        cmpi.w    #$0a,d1
```

near the top of page 24 should be deleted ...

[ND] Yes. This code is not necessary. It was put there as a 'bug fix' which George and I have discussed by email. The bug in question is in the documentation and it states that error returns would have negate, zero or positive values depending on what caused the end of the edit.
When I read the docs, I assumed, incorrectly, that this meant D0 would be set as above, however as George pointed out to me, it never mentions D0 and simply says that the Condition Codes would be set.
Because I assumed a bug, I wrote my library code to return a correct (ahem!) value in D0 which was what I though that the docs were telling me. They were not, it was my own inability to read English that was faulty.
As George says, those 4 lines can be removed. However, doing so will require changes elsewhere in the code. If those 4 lines are left in, then the comments in the header for IW_PRINT will be correct – and D0 will be set as advised.

[GG] 3b. ... as well as

```
        tst.l     d0
```

which appears three instructions later. My reason is this. All uses of iw_input in Lib Gen are in the three routines sym_hit, lib_hit and bin_hit. These are followed by a branch, if error, to the exit and later, by a call to iw_print. If the branch to the exit is taken it is because there is an error with the negative code being in D0. If there is no error then iw_print will almost immediately set D0 as a result of the call to wm_swinf. This means that there is no need to set D0 and test it in iw_input since this will have no effect.

[ND] The above is valid, in the current context of LibGen at least. I use this code in other programs. What you are seeing here is not the full library either, but a slightly reduced version to avoid having to print (or type in) code that is never called by this utility.

[GG] 4. In both lib_hit and bin_hit the routine ends with

```
        bsr       iw_print
        rts
```

In both cases this can be replaced by the single instruction

```
        bra       iw_print
```

Of course in that case the branches to bf_exit and lf_exit will have nowhere to go. These branches can be replaced, in both cases, by

```
        bra       sh_exit
```

[ND]. I try to keep my sub-routines/library stuff as 'one way in, one way out' Sometimes I fail in this, but usually, it's only possible to exit a library or subroutine by a single RTS. At my age, trying to keep track of multiple entries and exits is too much for my brain!

[GG] 5a. In li_avail on page 27 there should be a plus sign, not an underscore after ws_litem

[ND] Given that I've been executing this code quite happily, and so far, all is working as expected, I was confused by this. It appears to be a transcription error - which confuses me even more, as I import my code directly into the article! It is indeed supposed to be a plus sign. I have no idea where the underscore came from.
To be clear, it should look like this:

```
li_avail move.l     #$01011101,ws_litem+li_libfile(a1)
```

[GG] 5b. Also the final two instructions in this routine could be replaced by

```
        jmp        wm_ldraw(a2)
```

[ND] They could indeed.

[GG] 6. I must now apologise for returning once again to the 'error returns' from wm_ename. A close examination of the source code shows that, as is absolutely normal, the register D0 is either set to zero, showing no error, or is set to the negative error code. If there is no error, register D1.W will contain the terminating character, which must be one of ENTER, ESC, UP and DOWN.
What is also completely standard is that the condition codes are set to N if there is an error. What is unusual in this vector is that if there is no error the condition codes are set as the result of comparing D1.W with the code for ENTER ($0A). In the case of normal vectors you can branch on error using BMI or BNE. In this case you can't use BNE. The simplest way of alerting programmers to this fact is to tell them that condition code >0 can occur as well as ZERO when there is no error.
There is therefore no bug in wm_ename as Norman suggests on page 22.

Perhaps the statement in the manual, which is

'*Error returns*

Any I/O sub system errors

>0 if terminating character not <NL>'

is ambiguously succinct and might lead one to think that D0 as well as the condition codes is set greater than 0 for a terminating character other than ENTER. However, a later sentence in the manual reads
'*The condition codes will be set to -ve for an IO error, zero for ENTER or positive for other terminator.*'

[ND] Yes, George and I discussed this by email a while back. The splitting of my article into two separate parts gave us plenty of time to discuss the 'bug' and we agreed in the end that it was me who was wrong and the manual was correct.
As I explained above, my assumption that all the Error Returns in the QDOS/QPTR docs means D0 was incorrect and caused me the problems. I should learn to read and understand English I think. The manual talks of the Condition Codes and doesn't mention D0. I read that as meaning D0 rather than the condition codes.
As mentioned above, the next (exciting) instalment will have a correction noted.

[GG] 7. It is a very minor point but I wonder why Norman uses BLT instead of BMI to branch on an error. The instruction BLT branches if either one or the other, but not both, of N and V is set. BMI branches if N is set. Thus, if both N and V are set BMI will correctly cause a branch but BLT will not. It is unlikely, but nevertheless remotely possible, that both N and V could be set in a program.
The instructions

```
        moveq      #-1,d0
        and        #2,ccr
```

would do it.

[ND] Both BMI and BLT are signed comparisons, as George probably knows better than me. I often wonder why we need both of these though! BLT is used after a call to IW_INPUT which, as we know,

in its current state, is handling returns by setting D0 to negative, zero or positive and then setting the flags accordingly.
I admit, I've had to check my Motorola manual to check out the difference and to be honest, I suspect that I should be using BMI as it does indeed only check the N flag. BLT will take the branch if (N=1 and V=0) or (N=0 and V=1), so could take a branch on overflow only - which is unlikely in this situation, but in other programs, could be the cause of a hard to find bug.
Interestingly, Andy Pennell, in his book Assembly Language Programming, doesn't mention BLT on page 26 but does, later on in the book, on page 30 - but without giving a good explanation for their purpose. Hmmmm, perhaps the Motorola CPU has too many branching tests!

[GG] Finally I look forward eagerly to the next instalment.

[ND] There should be one somewhere else in this edition!

[ED] Yes - it is. Quite a long discussion this time considering the initial 'minor' ;-)

Directories

by David Denham

What is a directory?

Think of your computer as a filing cabinet with several drawers. Each drawer is one of your drives. Within each drawer you can have several folders each holding groups of files. The term 'directory' is synonymous with the term 'folder' as far as we are concerned.

Within each drive you can have several folders each holding groups of files. For example, one could contain all your letters, another could hold all your spreadsheets, while another could hold your SuperBASIC programs.

Each folder or directory can contain further folders. For example, the one containing your SuperBASIC programs can be further split into Games, Utilities, Music programs and so on. Directories within directories like this are referred to as Sub-Directories.

On an original QL, all files were just lumped together without any real form of grouping. This wasn't too bad when all we had was microdrive cartridges holding about 100 kilobytes each. Then we got floppy disks which could potentially hold dozens if not hundreds of small files. As we filled the disks up, a DIR command could give a very long list of files making it harder and harder to find the file we want. Imagine a disk containing over a hundred files - such long lists of files quickly become unmanageable!

When hard disk systems came along for QLs things got even worse. Now you could probably save hundreds if not thousands of files - something had to change.

And change it did. Add-ons like the Miracle Systems hard disc drive introduced us to Level 2 Filing Systems for the QL. The term Level 2 simply means the next level of QL filing systems - directories. The disc could be split into separate folders allowing you to group files together to make file management a lot easier. Such directories were also called 'hard directories'. The name came about since the original 'soft directories' were just files with a prefix added to the name!

How Do I Know Which Systems Handle Directories?

First you have to know that your computer or emulator is able to handle these Level 2 Directories. If it has a MAKE_DIR command built in, chances are you can use directories. If the manual doesn't tell you, check this with the EXTRAS command - the list should contain the MAKE_DIR command, or in some cases an equivalent function called FMAKE_DIR.

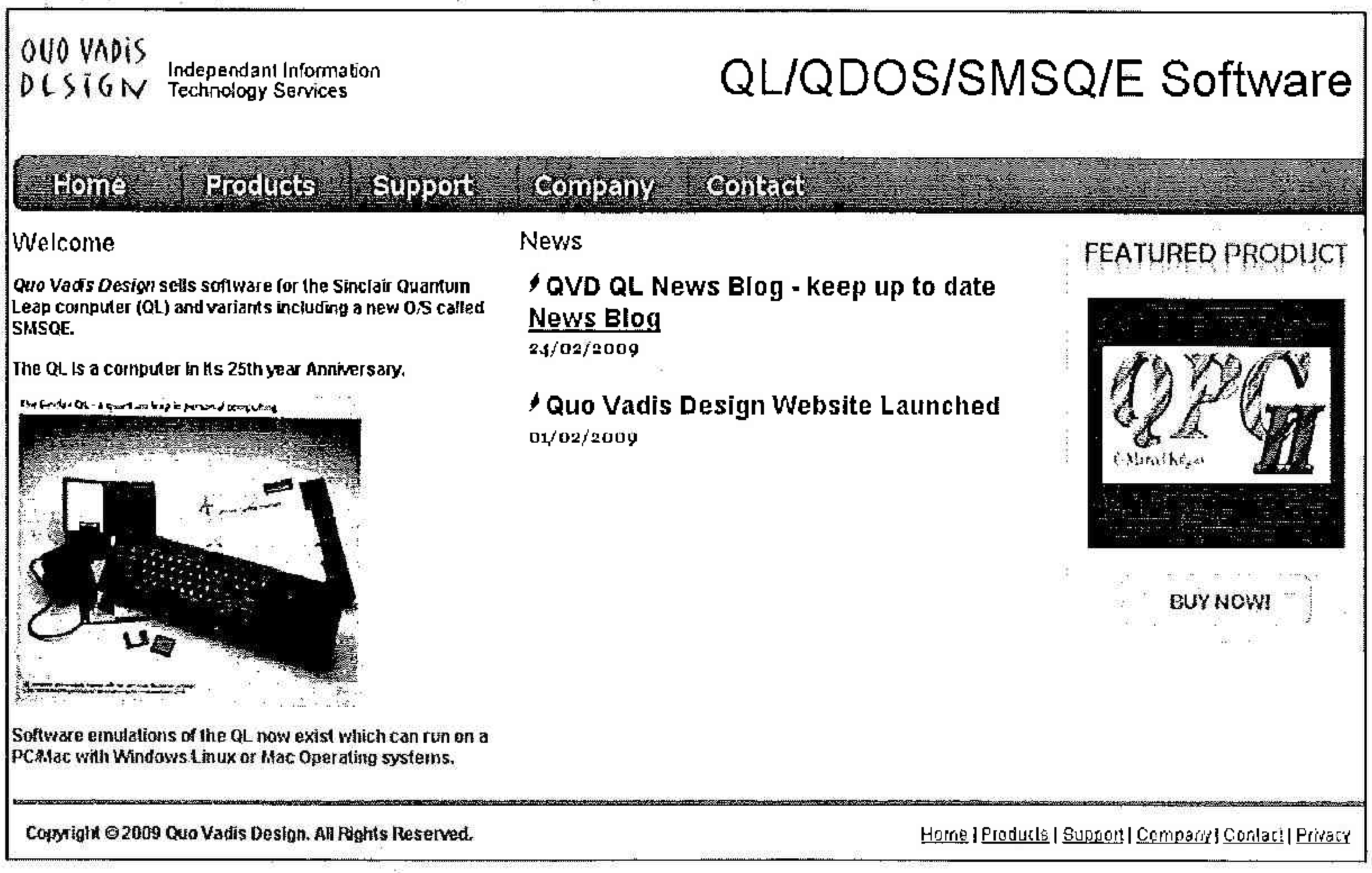
The situation is perhaps not as simple as we might like though.

QUO VADIS DESIGN

Independant Information Technology Services

www.ql-qvd.com

QL/QDOS/SMSQ/E Software


QUO VADIS DESIGN Independant Information Technology Services

QL/QDOS/SMSQ/E Software

Welcome

Quo Vadis Design sells software for the Sinclair Quantum Leap computer (QL) and variants including a new O/S called SMSQE.

The QL is a computer in its 25th year Anniversary.

Software emulations of the QL now exist which can run on a PC/Mac with Windows Linux or Mac Operating systems.

News

QVD QL News Blog - keep up to date
News Blog
24/02/2009

Quo Vadis Design Website Launched
01/02/2009

FEATURED PRODUCT

Copyright ©2009 Quo Vadis Design. All Rights Reserved.

Home | Products | Support | Company | Contact | Privacy

Special Offers available from Jochen Merz Software for its 25 years in QL Trading

Check the QL News Blog on our website for the special offers

Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

QL Today — The QL magazine for all QDOS, QL, SMSQ ... users!

Subscriptions taken online

An original Trump Card doesn't support level 2 directories or 'hard' directories, while later versions supplied by Qubbesoft had an updated ROM allowing use of directories! The table below lists some types of QL systems and whether or not they can support directories. The list is far from complete - if it's not in the list I just don't know!

System Type	Level 2 Directories?
Unexpanded QL	No
Microdrives	No
QL and Trump Card	Later versions only
QL and Gold Card/Super Gold Card	Yes
Other floppy disk systems	Mostly no, although updated Level 2 ROM could be fitted to some disk interfaces
Aurora	Depends on disk card interface fitted
Thor	Some versions - yes, although non-standard
Q40/Q60	Yes
Qubide	Yes
Miracle Systems Hard Disk	Yes
QLay and QL2K	No
Qemulator	Registered version only
QPC	Yes
SMSQmulater	Yes

Here is a simple little test routine which will just test for the presence of the MAKE_DIR or FMAKE_DIR extensions on your system. It needs a ramdisc for the temporary file.

```
1000 DEFine PROCedure Level2_Test
1010  LOCal testing,extension$
1020  OPEN_NEW #3,ram1_tmp
1030  EXTRAS #3
1040  CLOSE #3
1050  OPEN_IN #3,ram1_tmp
1060  REPeat testing
1070   IF EOF(#3)=1 THEN EXIT testing
1080    INPUT #3,extension$
1090    IF extension$ == 'MAKE_DIR' THEN
1100     PRINT'MAKE_DIR command present'
1110    END IF
1120    IF extension$ == 'FMAKE_DIR' THEN
1130     PRINT'FMAKE_DIR function present'
1140    END IF
1150  END REPeat testing
1160  CLOSE #3
1170  DELETE ram1_tmp
1180 END DEFine Level2_Test
```

Just enter the command LEVEL2_TEST and it will look through the list of extensions for MAKE_DIR or FMAKE_DIR, printing the name of whichever it encounters in the list.

If you prefer a function to test if there's MAKE_DIR or FMAKE_DIR on the system, try this: PRINT IsItLevel2 will return 1 if there is a MAKE_DIR or FMAKE_DIR extension.

```
2000 DEFine FuNction IsItLevel2
2010  LOCal testing,extension$
2020  OPEN_NEW #3,ram1_tmp
2030  EXTRAS #3
2040 CLOSE #3
```

```
2050  OPEN_IN #3,ram1_tmp
2060  lev2=0
2070  REPeat testing
2080   IF EOF(#3)=1 THEN EXIT testing
2090   INPUT #3,extension$
2100   IF extension$ == 'MAKE_DIR' OR extension$ == 'FMAKE_DIR' THEN
2110     lev2 = 1 : EXIT testing
2120   END IF
2130  END REPeat testing
2140  CLOSE #3
2150  DELETE ram1_tmp
2160  RETurn lev2
2170 END DEFine IsItLevel2
```

Creating Directories

Directories are created with the MAKE_DIR command like this:

```
MAKE_DIR drive_directory_
```

So to make a directory on FLP1_ to hold BASIC programs we use:

```
MAKE_DIR FLP1_BASIC_
```

If the system has the FMAKE_DIR function, it makes it easier to cope with error trapping:

```
LET errorcode=FMAKE_DIR(FLP1_BASIC_)
```

FMAKE_DIR will return a negative error code if something went wrong. This error code may be one of the following:

-7 = Not Found (drive not available)
-8 = Already Exists (directory of that name exists already)
-9 = In Use (directory of that name exists already)
-15 = Bad Parameter (cannot handle directories on this drive)

Some Rules

A few rules to note:

1. The directory name should end with an underscore character, although many systems will add this automatically if it is missing.
2. The name of the drive and directory does not have to be in quotes unless it contains a non-ascii character such as a full stop or space. The name can always be in quotes if you prefer, or may also be a string variable, e.g. LET d$="Flp1_Games_":MAKE_DIR d$
3. Most systems will let you include an underscore within the directory name, but this is poor practice. An underscore may mean the end of a directory name, so on some systems it may cause a mild confusion as to where the directory name ends. I tend to steer clear of underscore characters within a directory name. It makes life easier if you just use letters and numbers.

   Most systems will allow spaces if you want to make long name more readable, but of course the name has to be in quotes. I find it best to avoid spaces and other non-alphanumeric characters.
4. Name lengths. The sum length of both the directory name and a file name may not exceed 36 characters. So it is best to keep directory names short and meaningful, especially when using sub-directories within other directories, remembering that the underscore characters also count towards the maximum length of 36.
5. Like QL filenames, directory names are usually case insensitive – it doesn't matter if you use upper case, lower case or a mix. The only exception is some emulators which are subject to the case sensitivity of the file system on which they run. Linux, for example.

Saving Files To Directories

To save a file to a directory, we just include the directory name before the filename. We can save a BASIC program called mygame_bas into a directory called BASIC on drive WIN1_

```
SAVE win1_Basic_MyGame_bas
```

or, of course

```
SAVE "win1_Basic_MyGame_bas"
```

Copying a file from floppy disc to hard disc is just as simple:

```
COPY Flp1_MyGame_bas TO win1_Basic_MyGame_bas
```

If your system has the WCOPY command, you can use this to bulk copy files, e.g. to copy all files from Flp1_ to Win1_Basic_ just use:

```
WCOPY Flp1_ TO Win1_Basic_
```

Equally, if you wanted to copy files back from a hard disc directory back to a floppy disc:

```
WCOPY Win1_Basic_ TO FLP1_
```

What the command does is try to work out the differences between where the files are stored and copied to and change the full filename accordingly.

WCOPY can also use a simple form of wild card name, so that only files matching that wild card are copied. Supposing we have a disk full of all sorts of files and we wished to copy just the Quill documents to a directory called Win1_Quill_ and just the Abacus spreadsheets to Win1_Abacus_

```
WCOPY Flp1__doc TO Win1_Quill__doc
WCOPY Flp1__aba TO Win1_Abacus__doc
```

The wildcard is the extension - anything ending with _doc or _aba in these examples.

DIR List

When you DIR a disc containing directories, the name of a directory is shown on most systems by having the "->" characters after a name.

```
DIR WIN1_
```

might give

```
WIN1 QDOS 35840/40960 sectors
boot
Basic ->
Games ->
Xchange ->
```

In this example, I have a boot program written in Superbasic plus three directories called Basic, Games and Xchange. Note how the directories have -> after the name, and how the last '_' is not normally shown.

It is quite normal not to put important system files such as a boot program not into any directory. This is called putting a file in the 'root directory' or 'base directory'. You may also come across some programs which may not work correctly from a directory, so these can be put in a root directory if required as long as you remember to keep the number of files down to reduce clutter.

Sub-Directories

If we want to create sub-directories within an existing directory, that's quite easy. Just work out what you want and make sure the main directory exists first. In an example, I'm going to create a directory to hold a copy of Xchange, and within that four subdirectories to hold Quill, Abacus, Easel and Archive files.

First, I create the main directory called Xchange:

```
MAKE_DIR Win1_Xchange_
```

Next, I create the four sub-directories to go inside the Xchange directory, simply by adding the names of the sub-directories to the name of the main one. I'll call these sub-directories Quill_, Abacus_, Archive_ and Easel_

```
MAKE_DIR Win1_Xchange_Abacus_
MAKE_DIR Win1_Xchange_Archive_
MAKE_DIR Win1_Xchange_Easel_
MAKE_DIR Win1_Xchange_Quill_
```

I happened to put those in alphabetical order, but there is no real need to do this unless you want them to appear in alphabetical order in a DIR list.

If I now do a DIR of Win1_ the list is still shown the same as the first example above. But if I now do a DIR of Win1_Xchange_ I see the new sub-directories:

```
WIN1 QDOS
35840/40960 sectors
Xchange_Abacus ->
Xchange_Archive ->
Xchange_Easel ->
Xchange_Quill ->
```

So you can see that if you use DIR to list the content of a directory, it shows what's in that directory and not anything in the main directory before it. This brings us to their hierarchy – the first directory is referred to as the 'top level' directory.

When you have directories within directories, you can think of the whole structure as a 'tree', with various branches leading away from the root or top level. You can draw a diagram a bit like a family tree (an upside down tree) like this. This will help us envisage what a directory structure is like when it comes to 'navigation'.

```
              WIN1_
              |
              |
--------------------------------------------------
|             |             |
Basic_        Xchange_      Games_
              |
              |
--------------------------------------------------------
|             |             |             |
Abacus_       Archive_      Easel_        Quill_
```

Sub-directories within other directories are sometimes referred to as 'nested directories'.

Deleting Directories

From time to time we may need to delete a directory, either because we have finished using it, or we have realised we made a typing mistake and misnamed it. Suppose I'd intended to create a directory called Win1_Toys_ and accidentally typed MAKE_DIR Win1_Tyos_. The directory can be deleted with a simple DELETE command as long as you take care to type the name correctly to avoid accidentally deleting a similarly named directory!

```
DELETE Win1_Tyos_
```

However, you cannot normally delete a directory name if it already contains some files. You would need to delete those files first. So if I had a directory named Win1_Texts_ which contained three plain text files, I'd need to delete all three of those files before I could delete the directory called

Win1_Texts_. A quick but potentially dangerous way of deleting them would be to use the WDEL (Wildcard Delete) command to delete all its files first:

```
WDEL Win1_Texts_
```

then DELETE Win1_Texts_

If you have sub-directories within a directory, you will need to delete the content of the deepest directories first, then delete the sub-directory names, then delete all the files in the parent or higher directory and work your way backward until you reach the one you want to remove. Taking Win1_Xchange_ in the example tree above, in order to remove Win1_Xchange_ you'd need to delete the content of all four sub-directories first, then delete all four sub-directories, then delete any remaining files in Win1_Xchange_ before finally removing Win1_Xchange_. It is a bit tedious with a lot to type in, but perhaps that's no bad thing as it forces you to do it step by step to avoid accidentally deleting everything!

The sequence of BASIC commands you would need to delete Win1_Xchange_ above would be:

1. Remove sub-directory content:

```
WDEL Win1_Xchange_Abacus_
WDEL Win1_Xchange_Archive_
WDEL Win1_Xchange_Easel_
WDEL Win1_Xchange_Quill_
```

2. Delete sub-directories:

```
DELETE Win1_Xchange_Abacus_
DELETE Win1_Xchange_Archive_
DELETE Win1_Xchange_Easel_
DELETE Win1_Xchange_Quill_
```

3. Delete any other files left in Win1_Xchange_:

```
WDEL Win1_Xchange_
```

4. Finally delete Win1_Xchange_ itself:

```
DELETE Win1_Xchange_
```

Default Directories

It can be a bit tedious typing in very long drive and directory names, so Toolkit 2 added commands to set default directories. There are three of them in all – one called PROG_USE to set default directories for programs you can execute (e.g. Xchange or QL Quill), DATA_USE for ordinary data files and BASIC programs and DEST_USE for the default destination when copying files.

1. PROG_USE drive_directory_

Suppose you keep your executable programs in a directory called PROGS_ on Win1_. You can set PROG_USE Win1_Progs_ to make EXEC and EX commands start the programs from there by default without having to add a drive and directory. Let us compare two EX commands:

`EX Win1_Games_MyGame_bin` – this starts the game called MyGame_bin from Win1_Games_, in other words you specified a drive and directory called Win1_Games_

`EX Utility_Task` – here you have specified no drive and directory, so it looks on the one specified in the last PROG_USE statement automatically (called the Program Default Directory), in this case Win1_Progs_. Saves a bit of typing when you have grouped all of your programs in the same place.

2. DATA_USE drive_directory_

Suppose you keep your BASIC programs in Win1_Basic_. After DATA_USE Win1_Basic_ the computer will look for a Basic program or indeed any data file in this directory unless you explicitly state where:

`LOAD Win1_Xchange_MyBasicProg_bas` – this one specifies a drive and directory, so it will look for the file there.

`LOAD MyBasicProg_bas` - this doesn't specify a drive or directory, so it checks the DATA_USE default value. In other words, it realises you didn't specify anything, looks up the default (called the Data Default Directory) which was Win1_Basic_ so ends up loading the file from there.

Given that data files are more likely to be spread across many directories, this is less useful in this particular context, but another use is for the DIR command. If you had set DATA_USE WIN1_ then enter the command DIR with no drive name, it looks up the Data Default Directory value (Win1_) and tries to do a DIR of that, in this case like a DIR WIN1_.

3. DEST_USE drive_directory_

Sets the default destination directory when files are being copied, moved or renamed, usually when these commands which would otherwise be used with two names this provides the second or destination name. If the destination name does not end with a '_' it usually means it's a non-directory device, such as a serial port or parallel printer port.

This can be useful. Set DEST_USE 'PAR' on a system with a parallel printer port and this can be useful for copying files to a printer, using either COPY or SPL (spooler) commands:

`DEST_USE PAR`
`COPY win1_plain_txt` (sends it to PAR printer port)
`SPL win1_PLAIN_txt` (uses the SPL spooler job to print a file to a PAR printer port in the background).

I tend to find it best to group all my executable programs into one directory and point PROG_USE to that one. I also set DATA_USE to point to my main drive, WIN1_. DEST_USE is just used for the default printer port. My boot program contains lines like this:

```
200 PROG_USE Win1_Progs_
210 DATA_USE Win1_
220 DEST_USE Par
```

Many systems impose a limit of 32 characters on a default name. This is in keeping with the maximum filename length of 36 characters. After all, if you have a long 32 character default name, it only leaves 4 characters for the rest of the filename, so you can understand why!

Checking The Settings

Sometimes you may forget what defaults you have set. This is where the three functions DATAD$, PROGD$ and DESTD$ come in. These request and return the relevant information from the system:

`PRINT PROGD$` returns the default set with PROG_USE
`PRINT DATAD$` returns the default set with DATA_USE
`PRINT DESTD$` returns the default set with DEST_USE
An additional command `DLIST` prints a list of all three settings.

Using these extensions you'll quickly realise that the system can only have one default for the entire computer. In other words, when you set PROG_USE 'Win1_' it applies to all programs. It is not possible for each program to have a separate setting. Also, not all programs understand these defaults - life is not always easy! In a future article I'll show how to work around these limitations for awkward programs.

Default Defaults

When you first start a QL or emulator, these defaults are usually set to one of the drives in the system and no directory. If your system has only floppy disc drives and no ard disc drive, it will usually set PROG_USE to FLP1_ and DATA_USE to FLP1_ (on some systems with two disc drives DATA_USE may be set to FLP2_), and DESTD$ is usually a serial port or printer device such as PAR or SER1

Navigation

Some extensions are usually provided to navigate the defaults up and down a directory tree. Personally, I rarely (if ever) find the need to use these, but they are there. Normally, these three commands work on the DATA_USE default name, but if the PROG_USE name happens to be the same they will affect that too.

DDOWN name – moves down the tree, adds the name given to the default, so if the default happened to be Win1_ and the command was DDOWN Xchange the default would now become Win1_Xchange_. A subsequent DDOWN Quill would change the default to Win1_Xchange_Quill_

DUP – moves up the tree by removing the last directory name, so if the default was set to Win1_Xchange_Abacus_, a DUP would make it Win1_Xchange_ and a second DUP would make it just Win1_

DNEXT – moves up and then down a different part of the directory tree.

Separating Directory And Filename

In some ways, the QL filing system is a bit restrictive. Apart from the 36 character total name length limit, it's also hard for a program to work out which part of a name is a directory and which is the filename part.

Most operating systems use specific symbols in a filename to separate the directory and filename, for example on Windows we might save something to C:\QLfiles\letter.txt. Here we know that the drive name is C:, the directory name is indicated by the backslash characters (directory called QLfiles) and the filename is the bit after that last backslash, here "letter.txt". The QL is a bit more simplistic in this respect and the directory separator character can also be part of the filename, potentially making it hard to work out which is the directory name part, and which is the filename part.

There is a solution, but it's not obvious and requires a little bit of devious code to ask the system which is the directory, then subtract from the full filename (or full 'path' name as it's sometimes referred to). The answer lies in the use of the FNAME$ function, which returns the name of a file normally, or of the directory partif you open a channel using the FOP_DIR command:

```
LET f$="Win1_Xchange_test_doc"
LET channel=FOP_DIR(f$)
PRINT FNAME$(#channel)
CLOSE #channel
```

This prints Xchange, the name of the directory containing the file called TEST_doc. Note that it doesn't return Xchange_ only the part before the underscore, so you need to take that into account. Also, not including the drive name.

Separating the directory path and the filename needs a bit of juggling like this (I wrote an article about Directory Names back in Volume 10 Issue 4 of QL Today).

First, we use FNAME$ to return the full path and filename without the drive name:

```
LET f$="Win1_Xchange_test_doc"
LET ch=FOP_IN(f$)
LET fullname$=FNAME$(#ch)
CLOSE #ch
LET ch=FOP_DIR(f$)
LET dirname$=FNAME$(#ch)
IF dirname$<>"" THEN LET dirname$=dirname$&'_'
CLOSE #ch
LET filename$=fullname$(LEN(dirname$)+1 TO LEN(fullname$) #
PRINT filename$
```

which (eventually!) prints 'test_doc'

Conclusion

Directories are a very useful and sometimes underused facility on modern QL systems. They take a bit of getting used to but once you've got used to them you'll wonder how you ever managed without them. The main advice is to remember that the length of a directory name and filename combined must not exceed 36 characters and try to plan ahead what names you intend to use because if you change your mind later it can be a very fiddly job to make big changes.

QUANTA

QL USERS AND TINKERERS ASSOCIATION

Next QUANTA Sponsored Event

Annual General Meeting 2013 And Workshop, Date: Sat/Sun 13th/14th April

Opening on Saturday at 11.00 am (Doors open 10 am for setting up),

Sunday at 10.00 am with the AGM at 2.00 pm in the afternoon.

The Venue will be in Manchester at the

3rd Davyhulme Scout Headquarters, "The Endeavour", Conway Road, off Lostock Road, Davyhulme, Manchester. M41 0TF. Near M60 J9.

Ask for our special 3 Year discount

Further details from the Membership Secretary
John Gilpin, 181, Urmston Lane,Stretford
Manchester. M32 9EH (UK)
Tel. 0161 865 2872

Email: membership@quanta.org.uk
Website: http://www.quanta.org.uk

I2C Interface for QL Emulators Part 6 - (Correction)

by Ian Burkinshaw

Some of you may have noticed my deliberate mistake in the last issue of QLToday. I did not include the software listing!!!! Completely my fault. So here are the missing listings.

See the I2C Interface article listing in QLT Vol 17 issue 1 page 40. You need lines 1000 to 3090 from this listing. Then merge it with the following listings.

```
10 REMark I2C test routines
20 I2C_init
30 I2C_Start
40 :
50 MCP_init
60 PRINT
70 PRINT "LED Flash (Port A)"
80 ledflash parallel8$,"A"
90 PRINT
100 PRINT "LED Flash (Port B)"
110 ledflash parallel8$,"B"
120 PRINT
130 PRINT "LED Flash (Port A)"
140 ledflash parallel7$,"A"
150 PRINT
160 PRINT "LED Flash (Port B)"
170 ledflash parallel7$,"B"
180 PRINT
190 PRINT "All LED Flash"
200 allledflash
210 PRINT
220 PRINT "LED Binary Count (Port A)"
230 ledcount parallel8$,"A"
240 PRINT
250 PRINT "LED Binary Count (Port B)"
260 ledcount parallel8$,"B"
270 PRINT
280 PRINT "LED Binary Count (Port A)"
290 ledcount parallel7$,"A"
300 PRINT
310 PRINT "LED Binary Count (Port B)"
320 ledcount parallel7$,"B"
330 PRINT
332 PRINT "LED Chase"
335 ledchase
340 :
350 PRINT "End         ":CLOSE#3:STOP
360 :
500 DEFine PROCedure monitor
510 c$=""
520 REPeat test
530 a$=INKEY$(#3)
540 IF a$="" THEN GO TO 530
550 c$=c$&a$
560 END REPeat test
570 END DEFine monitor
580 :
```

Lines 1000 to 3090 from QLT 17

```
4000 DEFine PROCedure ledflash(dadd$,port$)
4010 FOR a=1 TO 10
4020 IF port$="A" THEN PRINT#3;"s-";dadd$;" 12 ff p";CHR$(13);:REMark s=start message to USB to
     I2C converter, p=end of message to USB to I2C converter.
4030 IF port$="B" THEN PRINT#3;"s-";dadd$;" 13 ff p";CHR$(13);:REMark s=start message to USB to
     I2C converter, p=end of message to USB to I2C converter.
4040 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared
4050 PAUSE 25
4060 IF port$="A" THEN PRINT#3;"s-";dadd$;" 12 00 p";CHR$(13);
4070 IF port$="B" THEN PRINT#3;"s-";dadd$;" 13 00 p";CHR$(13);
4080 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared
4090 PAUSE 25
4100 NEXT a
4110 END DEFine ledflash
4120 :
5000 DEFine PROCedure ledcount(dadd$,port$)
5010 FOR a=0 TO 255
5020 hhex$=HEX$(a,8)
5030 PRINT a;" ";hhex$;" ";
5040 lhex$=hex_case_con(hhex$)
5050 IF port$="A" THEN PRINT#3;"s-";dadd$;" 12 "&lhex$&" p";CHR$(13);
5055 IF port$="B" THEN PRINT#3;"s-";dadd$;" 13 "&lhex$&" p";CHR$(13);
5060 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared
5070 PAUSE 5
5080 NEXT a
5090 END DEFine ledcount
5100 :
6000 DEFine PROCedure MCP_init
6010 PRINT#3;"s-";parallel8$;" 00 00 p";CHR$(13);:REMark set mcp23017 port "A" mode IODIRA to
     output
6020 non_print_reply
6030 PRINT#3;"s-";parallel8$;" 01 00 p";CHR$(13);:REMark set mcp23017 port "B" mode B to output
6040 non_print_reply
6050 PRINT#3;"s-";parallel7$;" 00 00 p";CHR$(13);:REMark set mcp23017 port "A" mode IODIRA to
     output
6060 non_print_reply
6070 PRINT#3;"s-";parallel7$;" 01 00 p";CHR$(13);:REMark set mcp23017 port "B" mode B to output
6080 non_print_reply
6090 END DEFine MCP_init
6100 :
6110 DEFine PROCedure allledflash
6120 PRINT#3;"A40";CHR$(13);
6130 non_print_reply
6140 FOR a=1 TO 10
6150 PRINT#3;"s-";parallel8$;" 12 ff p s-";parallel8$;" 13 ff p s-";parallel7$;" 12 ff p s-";
     parallel7$;" 13 ff p";CHR$(13);:REMark s=start message to USB to I2C converter, p=end of
     message to USB to I2C converter.
6160 REMark PRINT#3;"s 13 ff p";CHR$(13);:REMark s=start message to USB to I2C converter, p=end
     of message to USB to I2C converter.
6170 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared
6180 PAUSE 25
6190 PRINT#3;"s-";parallel8$;" 12 00 p s-";parallel8$;" 13 00 p s-";parallel7$;" 12 00 p s-";
     parallel7$;" 13 00 p";CHR$(13);:REMark s=start message to USB to I2C converter, p=end
     of message to USB to I2C converter.
6200 REMark PRINT#3;"s 13 00 p";CHR$(13);:REMark s=start message to USB to I2C converter, p=end
     of message to USB to I2C converter.
6210 non_print_reply:REMark Stops printing the USB to I2C reply also ensure serial buffer is cleared
6220 PAUSE 25
6230 NEXT a
6240 END DEFine allledflash
6250 :
6260 DEFine PROCedure ledchaseup
6290 FOR a=1,2,4,8,16,32,64,128,0
6300 PRINT#3;"s-";parallel8$;" 12 ";HEX$(a,8);" p";CHR$(13);
6310 non_print_reply
```

```
6320 PAUSE 1
6330 NEXT a
6340 FOR a=1,2,4,8,16,32,64,128,0
6350 PRINT#3;"s-";parallel8$;" 13 ";HEX$(a,8);" p";CHR$(13);
6360 non_print_reply
6370 PAUSE 1
6380 NEXT a
6390 FOR a=1,2,4,8,16,32,64,128,0
6400 PRINT#3;"s-";parallel7$;" 12 ";HEX$(a,8);" p";CHR$(13);
6410 non_print_reply
6420 PAUSE 1
6430 NEXT a
6440 FOR a=1,2,4,8,16,32,64,128,0
6450 PRINT#3;"s-";parallel7$;" 13 ";HEX$(a,8);" p";CHR$(13);
6460 non_print_reply
6470 PAUSE 1
6480 NEXT a
6490 END DEFine ledchaseup
6500 :
6510 DEFine PROCedure ledchasedown
6520 FOR a=128,64,32,16,8,4,2,1,0
6530 PRINT#3;"s-";parallel7$;" 13 ";HEX$(a,8);" p";CHR$(13);
6540 non_print_reply
6550 PAUSE 1
6560 NEXT a
6580 FOR a=128,64,32,16,8,4,2,1,0
6590 PRINT#3;"s-";parallel7$;" 12 ";HEX$(a,8);" p";CHR$(13);
6600 non_print_reply
6610 PAUSE 1
6620 NEXT a
6630 FOR a=128,64,32,16,8,4,2,1,0
6640 PRINT#3;"s-";parallel8$;" 13 ";HEX$(a,8);" p";CHR$(13);
6650 non_print_reply
6660 PAUSE 1
6670 NEXT a
6680 FOR a=128,64,32,16,8,4,2,1,0
6690 PRINT#3;"s-";parallel8$;" 12 ";HEX$(a,8);" p";CHR$(13);
6700 non_print_reply
6710 PAUSE 1
6720 NEXT a
6730 END DEFine ledchasedown
6740 :
6750 DEFine PROCedure ledchase
6760 PRINT#3;"s-";parallel8$;" 12 00 p s-";parallel8$;" 13 00 p s-";parallel7$;" 12 00 p s-";
     parallel7$;" 13 00 p";CHR$(13);:REMark s=start message to USB to I2C converter, p=end of
     message to USB to I2C converter.
6770 non_print_reply
6780 FOR cc=1 TO 10
6790 ledchaseup
6800 ledchasedown
6810 NEXT cc
6820 END DEFine ledchase
```

QL on a Linux Stick

by Norman Dunbar

Introduction

As many of you know, Dilwyn Jones has created a package of software that he distributes on a CD and you copy the contents to a USB stick this creating a QL on a Stick. It's relatively simple, and works, provided you are using a Windows computer of course.

Not everyone is, myself included, and Dilwyn has been asked on occasion if there is a Linux version available. Dilwyn is a self confessed Linux non-guru, and asked on the QL Users list for a volunteer, or two, to assist in the creation of QL on a Linux Stick. I volunteered 'a long time ago' and since then have failed to create anything!

How hard can it be?

Background

Ok, when you insert a USB stick, or CD into a Windows machine, the system takes over and mounts the device before looking for an autorun script that tells it what to do next. This is very insecure and Windows allows people to turn off the ability to auto run 'stuff' from a freshly mounted device. Not many people have turned it off, and probably, not many know about it. Microsoft tend to hide away security details from their users and set up weird and wonderful defaults that make the system totally insecure.

For example, the administrator password on XP is blank. It tells you that on page 16 of a tiny, meaningless, waffling 'manual'. My ZX-81 came with a far better manual!

Another thing Windows does in the background is this. When you install Windows, you install everything. You get all the support files for any program created with any of the Windows compiler or development tools. They are all there, hidden away. This means that Windows developers don't need to distribute the support files - dlls etc - with their application binaries. Microsoft made great noises about this when comparing their tools to those from other companies like Borland. The Microsoft tools created far smaller binaries. They neglected to add that the reason was because Windows already had everything you 'needed' - whether or not you needed it!

Linux is different. Yes there are lots of libraries installed - the equivalent of Windows dlls - but the only ones installed are those that are specifically required for any application you install. This makes sense as your hard disc isn't polluted with files you will never need, taking up space and time when you carry out a backup. You do backups don't you? ;-) So, when you install a new application on Linux, the installer (package manager actually) is told what libraries are required to install and run this particular application, and goes off to get them. It makes sense.

Security

As mentioned above, Windows is incredibly insecure. Everything is already installed, so a problem in a dll or device driver could easily be used to facilitate illegal access to your computer and either access your files, data, bank or personal details and so on. It can lead to your computer being used as part of a bot-net attack on some company or even, government. And all without you knowing about it.

Having CDs and USBs auto execute some nasty software is simple enough on a Windows machine. It's not impossible on Linux, but has a far lesser risk. There is, of course, autorun.sh on Linux CDs. These cause the CD to act like Windows and carry out some task after the device is mounted. Many desktop Linux distributions (aka distros) have auto-mount enabled so shoving a CD or USB will mount it for you automatically. But not all do this!

Problems Problems

So, in the Linux world where we have a more secure setup, we have three main problems in creating QL on a Linux Stick:

* Not everything is installed, so a program's dependencies might not be there when we insert the USB stick.
* Not all Linux distros will auto mount the USB stick on insertion.
* Not all Linux distros allow autorun.sh to execute on mounting a device.

Potential Solutions

I see two potential solutions to resolve the above. The first is to simple create a clone of the Windows version on QL on a Stick and let the user worry about dependencies etc. The huge problem with this method is simple, Dilwyn will get a huge number of emails saying 'it doesn't work' and they will then be passed on to me to sort out! This is a bad thing! (For both of us!)

The second is to build a Live distro and install all the dependencies required on that, and then use that on the CD that Dilwyn distributes to the users. They then copy that to their USB and off they go, everything is fine. Or is it?

The second option is the better of the two, but....

More Problems

A Linux live distro is one which runs directly from a CD or USB device. You plug it in, reboot and it boots up and loads the operating system from the device, rather than the hard disc, and it runs either from a 'squash-fs' system on the CD or from RAM.

Squash-fs is a compressed file system that allows much more data and applications to be squeezed onto a CD than the normal 650-700Mb would allow. It is decompressed on the fly as it is accessed, so it slows down the normal running of the system.

Some of the smaller distros copy themselves into RAM and run, very quickly, from there thus freeing up the CD to be used for other purposes - listening to music, burning backups and so on. Other distros run from the compressed file system on the CD, and run slower as a result. Sometimes very much slower as the CD has to be powered up, brought up to speed and then the data can be read - and it takes time.

The bonus to this type of Live distro is that you can 'try before you buy' and if you don't like it, simply don't use it, your computer remains untouched completely. Nothing is written to the hard disc, nothing is read from it. You remain, pure and safe.

So far, so good, but, unlike windows, you cannot simply copy the CD files onto a USB stick and boot from that, you need to install them correctly. Installing usually means that you login to the CD based system first and choose the option to install to USB device and work from that.

Windows users are happy in that a USB device is mounted and available as a E: drive, or similar.

Linux devices may be /dev/sdb or /dev/sdc and so on, depending on your system. And then there are partitions on the device - /dev/sda1, /dev/sda2 and so on. What are they all about?

Luckily, you don't need to worry about those but come installation to the USB device, you might need to consider them - you may be prompted to specify a partition size for swap space (your Windows virtual file), your root (/) partition and so forth. And of course, different live distros do a different partitioning scheme and use different utilities and so forth. Sigh!

No One Said it Would be Easy!

So far, I've managed to build a working Arch Linux, 32bit set up where on login, I get a menu with all the options to run any of the emulators, read the documentation or run the utilities. This is the most progress I have had.

I have a nice QL boot screen when you first run up the Linux system, and when the user chooses to boot (unfortunately not with F1 or F2) it automatically logs you in as the Qdos account where the menu is set to auto run. It works!

Unfortunately, the tools to create a live booting system from my installed development system appears to change things. My nice screens and my qdos user vanish and the menu doesn't auto run.

Frustrating? You bet!

Installing this (non-working) live distro to a USB stick, and booting it on my wife's laptop results in a pause looking for some device, then a drop into a rescue session. Not very desirable for people who might not understand Linux. Back to the drawing board.

I should add, the bootable USB version was created with a utility named 'Unetbootin'- it appears not to work. I have to use this utility because Arch Linux doesn't come with an installer as such, it's up to you, the Linux guru, to sort out installing an Arch system.

Maybe some other base distro would be better and a bit more user friendly? Well, I've looked at Linux Mint (an excellent distro if you are coming over from the 'dark side' for the first time) and unfortunately, I have been unable to follow the (outdated) wiki where the information is given on how to reorganise a Mint live distro to your own requirements and burn an iso file to boot from (or install on a USB) - but at least Mint comes with an installer.

* OpenSuse. I tried to go online and create a live distro. It too failed to work.
* Puppy Linux, supposedly easy to use and small. It runs totally from RAM. Well, on my laptop, it did boot quickly and run, but it was setting itself to a huge sized screen and the mouse pointer couldn't get to some of the options, icons and so on. Not good.
* Ubuntu. Nope, a respin here failed as well.

Sounds like the only people who appear to be able to create their own live distros, based on a 'production' one, are the same people who produce the production ones, doesn't it. Hmmm. Linux is excellent, but this doesn't make me happy!

Even More Problems

In the original QL on a Stick, Dilwyn supplied a few utilities. QStripper and WXQT2 for example. These are good tools (declaration of interest - QStripper is mine!). One of the first things I did was to replace the Windows version of QStripper with my own compiled for 32 bit Linux version. It runs as a native Linux program rather than a Wine emulated Windows program. (Although it runs fine either way!.)

WXQT2 on the other hand, is the Windows version and runs under wine. This is ok, and everything appears to work, until you try to get help. It pops up a message saying to enable an

The Sinclair QL Forum
A Place for Sinclair QL Users to Meet!
QL Forum
The online community for all things ql
www.qlforum.co.uk

View [illegible] e topics

GENERAL	TOPICS	POSTS	LAST POST
The Welcome Area Introduce yourself here!	23	163	by igorsteller Fri Feb 10, 2012 3:06 pm
Help for New Users A section for new users to ask for help and also for people to post helpful tips.	14	87	by olliraa Mon Feb 06, 2012 12:59 pm
General QL Chat A place to discuss general QL issues.	50	336	by [illegible] Fri Feb 10, 2012 3:31 pm
Hardware Nagging hardware related question? Post here!	70	589	by thorsinclair Fri Feb 10, 2012 12:35 pm
S[illegible]are A[illegible]	[illegible]	[illegible]	[illegible]
QL Emulation Discussion and advice about emulating the QL on other machines.	22	122	by Brane2 Sun Feb 05, 2012 10:23 pm
Compatibles Corr[illegible] An area to discuss [illegible] QL co[illegible] computers [illegible] as the Thor, Auror[illegible]	2	25	by [illegible] Fri Jan [illegible] 11:25 pm

MARKETPLACE	TOPICS	POSTS	LAST POST
For Sale Sell your QL items here!	19	81	by RWAP Thu Jan 19, 2012 5:26 pm

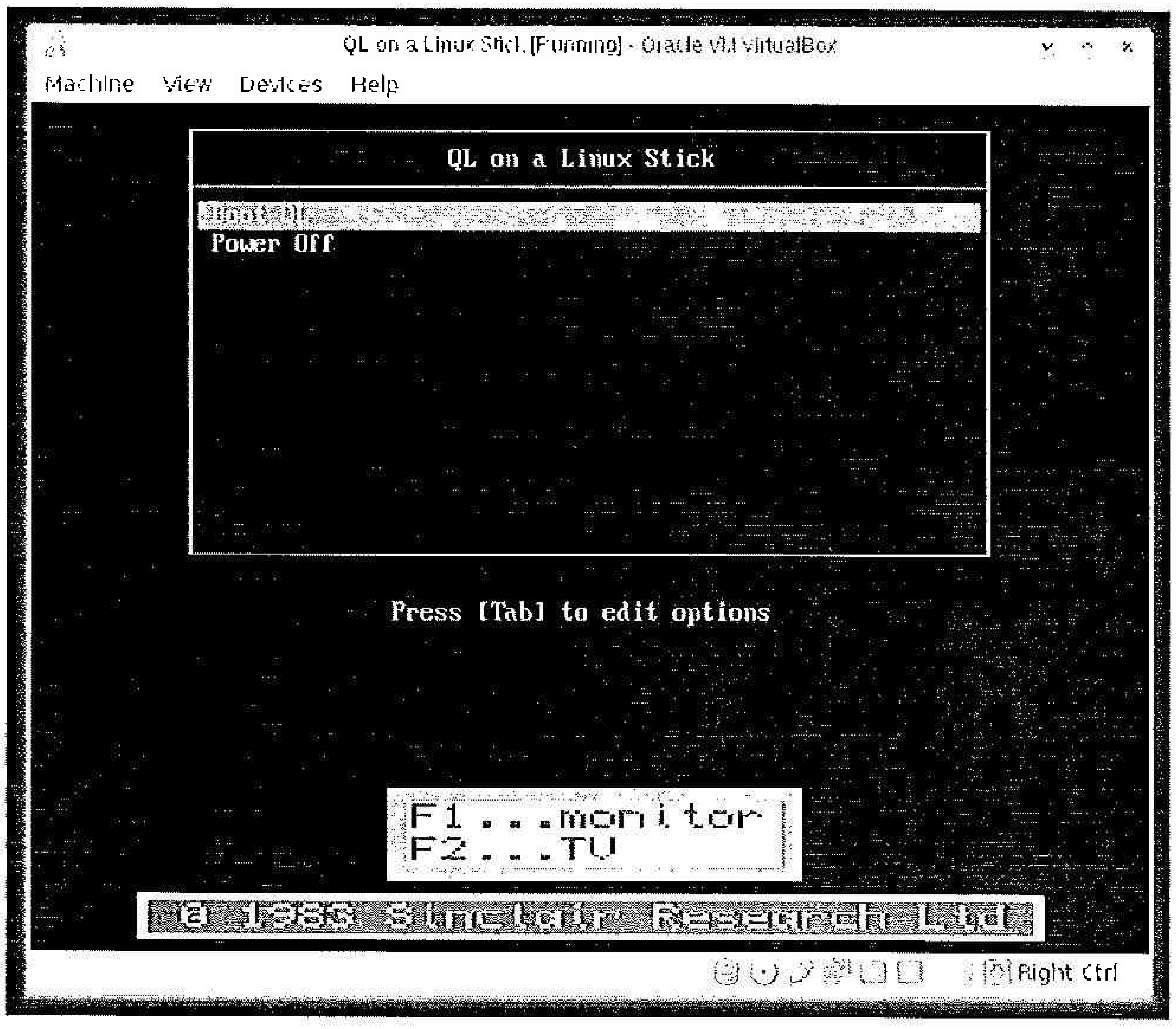
environment variable to enable help, and restart the program. Sounds easy but under wine, it doesn't work.

No worries, the code is supplied and it's built to be compiled under Windows, Mac or Linux. Except, the versions of the compiler and support tools have moved on a long way since this utility was first invented. There are huge numbers of errors that prevent a proper working Linux version from being created!

Searching online for a Linux version also turned up nothing. It's a utility that is distributed as source code and you have to compile it. So my next task is to dig into the source and try to work out, and fix, all those warnings and errors to get a working Linux version. Another problem to be resolved before we get QL on a Linux Stick.

Good News?

At least I have a project name - QL on a Linux Stick is a bit unwieldy, I'm calling it QLOLS for now! Here are a couple of screen shots showing the system as it is now. The first is the boot menu.


QL on a Linux Stick [Running] - Oracle VM VirtualBox

Machine View Devices Help

QL on a Linux Stick

Boot QL
Power Off

Press [Tab] to edit options

F1...monitor
F2...TV

© 1983 Sinclair Research Ltd

Right Ctrl

And the screenshot on the next page shows the system up and running with a copy of Wolfgang Lenerz's SQSQmulator running alongside the menu system that is presented on login.

If you look very closely at the task bar above, you will notice a small union flag - it's the third icon in from the left. By clicking on that flag, you can change the keyboard layout between almost any number of regional variations. I have it configured for UK, Germany, France, Belgium and so on at the moment, but it's simple to add others - US for example.

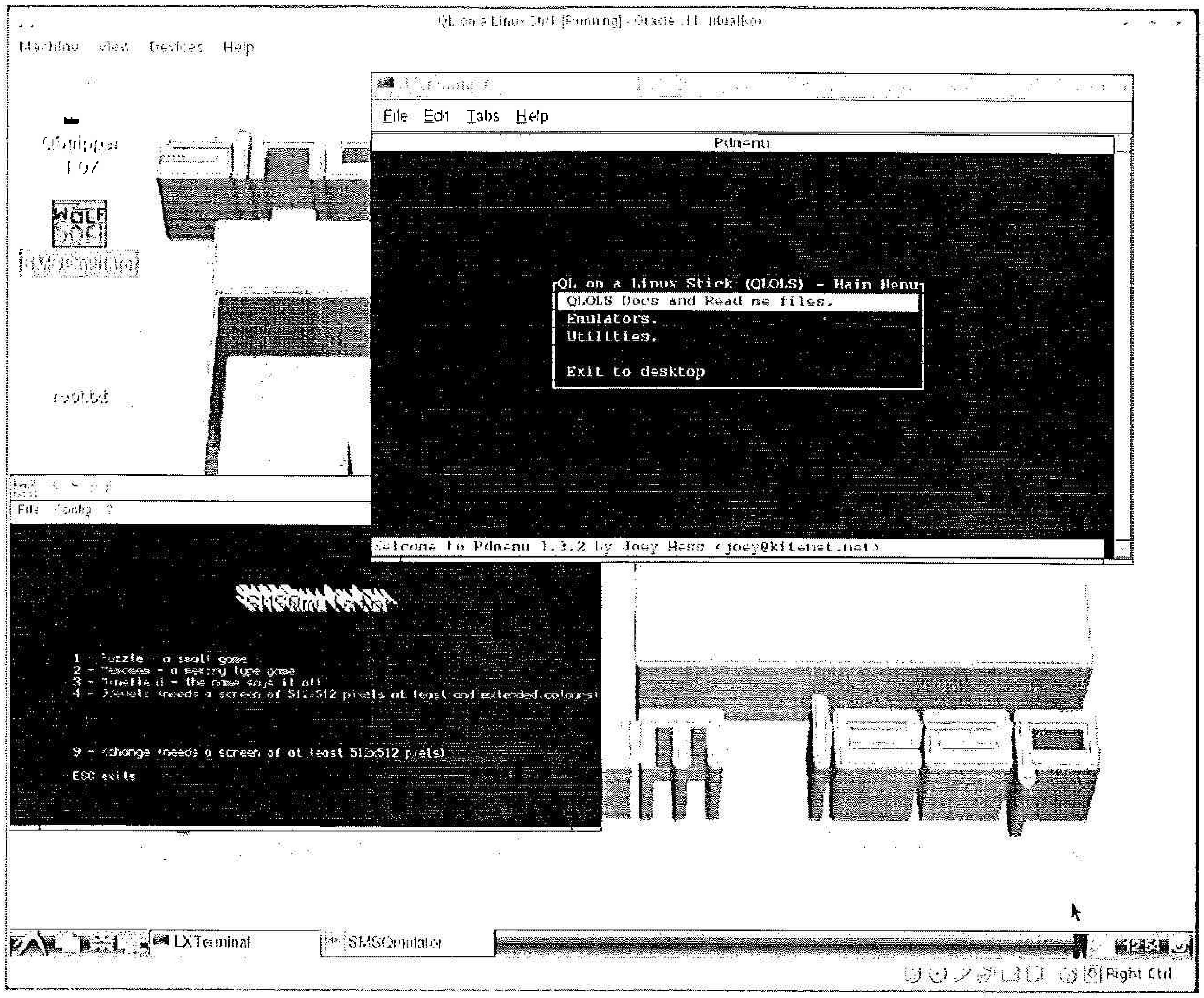
If Dilwyn looks closely at the image above, he might notice that the desktop wallpaper (do you really wallpaper a desktop? I don't! Surely it's a tablecloth!) is blatantly stolen from his own website's 'QL Logo' section.

And Finally!

Finally, if you are looking for a QLOLS system, there might be one day, when I resolve the above problems. But if you were asking why there isn't one, at least you know now!

If you have a properly compiling version of WXQT2, that compiles under GCC 4.6 and WxWidgets 2.8, then I'd be glad to have a copy, thanks. You'll save me a lot of work in getting the utility up to date.

That request applies to the dependencies for the utility, QLTools etc need also to be compiled for Linux. I'm pretty certain that running a Linux version of the GUI, talking to the Windows versions of the actual utilities, just isn't going to work!



```
QL on a Linux Stick (QLOLS) - Main Menu
 QLOLS Docs and Read me files.
 Emulators.
 Utilities.

 Exit to desktop
Welcome to Pdmenu 1.3.2 by Joey Hess <joey@kitenet.net>
```

```
1 - Puzzle - a small game
2 - Mexcess - a memory type game
3 - Tinetle it - the name says it all
4 - Qlevels (needs a screen of 512x512 pixels at least and extended colours)

9 - Xchange (needs a screen of at least 512x512 pixels)

ESC exits
```

Glossary of Abbreviations and Terms Part 7 - R to S

by Dilwyn Jones and Lee Privett

RAM	Random Access Memory, the memory used in the QL. You can read information from and write information to this type of computer memory. RAM is also the device name used by the QL for ramdisks. Usually up to 8 of these ramdisks can be used on a QL, with names ranging from RAM1_ to RAM8_
RAMdisk	A QL device for storing information in memory in a manner broadly similar to a floppy disk or Microdrive cartridge. Fast, but contents lost when you switch off or reset the QL. Useful for copying files to temporarily on a single drive computer, for example.
Recursion	A programming technique whereby a routine calls itself one or more times until a given condition is met. A very difficult concept for a beginner in programming to grasp!
Reference	See 'Pass By Value Or Reference' above.

Register	A part of a CPU which holds some binary value. May be an 8-bit value, 16-bit and so on. A value in a register can be processed much faster by a CPU than when that value is in a RAM location.
RISC	Reduced Instruction Set Computing, supposedly the fastest and most efficient way of computer processing
RGB	Red, Green, Blue. Three components of a colour video monitor signal
ROM	Read Only Memory, or memory which you can only read information from. Once information has been programmed into this type of memory, that's it, it can't be changed. The QL's operating system, a program or collection of small utilities which determines how the QL starts up and operates, is stored in this type of memory. ROM is also the device name for a RomDisq (q.v.) flash memory expansion card. You would use ROM1_ for the RomDisq like you would use MDV1_ for a microdrive, for example.
RomDisq	Flash memory card from TF Services. Plug into the EPROM slot behind the QL and contains from 2MB to 8MB of RAM. Uses the device name ROM1_
RS232C	I don't know what the letters stand for, but basically this is the name of the system used for sending data 1 bit at a time down a serial link such as the SER1 or SER2 sockets on the QL
RTC	Real Time Clock
RTM or RTFM	Something a trader or programmer is likely to tell you when you haven't read the instructions. Stands for Read The Manual, but I'll let you guess what the F in the second version stands for.
RTS	Request To Send, an RS232C signal pin
R/W	Read/Write. Getting information from or sending information to something
SATA	Serial Advanced Technology Attachment, an interface allowing fast read/write access to hard disk drives
SAV	Filename extension used for BASIC programs saved with the QSAVE command. This type of file is a tokenised BASIC program, as opposed to BASIC programs saved as plain text files with the ordinary SAVE command.
SB	SuperBASIC
SBASIC	An enhanced version of the QL's SuperBASIC, supplied with the SMSQ operating system. A variant of the name is S*BASIC, with the '*' acting as a wildcard character allowing the name to refer to either SuperBASIC or SBASIC (or sometimes both).
S*BASIC	Generic term used to refer to both SuperBASIC and SBASIC.
SCR	Screen window. A type of window on the display where you can PRINT information to. SCR windows have no keyboard facility, so you cannot use the INPUT command to allow the user to enter any information in that type of window
Scripting	Essentially, what we on the QL have thought of as 'programming'. The task of writing code in a high level language such as BASIC which is then interpreted by the computer into machine code as the program runs.

SD	Single Density – term used to refer to the recording 'density' of a floppy disk. Not often seen used with a QL. 'SD' also refers to a type of flash memory card, called a "Secure Digital" card.
SDUMP	The name of the Screen Dump software built into some systems, such as Gold Card and Trump Card. The term 'screen dump' refers to the act of printing a copy of the screen on paper, or less commonly the act of saving a copy of the screen as a file on disk.
SER	One of the serial ports on a QL. This is the name by which these sockets on your computer are known to the computer and to the software it runs. On a PC, the sockets might be called COM1: or COM2:, but the QL emulators such as QPC and QXL always refer to them as SER1 or SER2. SER is an abbreviation for SERIAL, which means that every bit of information sent to these ports is sent one after the other, in serial fashion, rather than say 8 bits at a time. See PAR above
SERNET	Software system used to control a group of computers (normally running the SMSQ/E operating system) connected together by serial port cabling.
Server	A computer which provides services used by other computers, e.g. the user might be using a program or web page on his computer which was stored on a remote server and sent to the user's computer to run or view.
SGC	Super Gold Card, an enhanced version of the Gold Card expansion card for the QL with 4MB of RAM and a 68020 processor.
Shriek	Alternative name used by some programmers for the exclamation mark symbol "!"
SIMM	Single Inline Memory Module, a type of memory card used by PCs
SMSQ	Operating system for the QXL, from Miracle Systems. Unlike SMSQ/E, this does not include the Pointer Environment. The letters SMS were never well defined, some say it stands for Single-user Multitasking System, while others say it stands for Small Microcomputer System, and others say Smart Micro System!
SMSQ/E	Extended version of the SMSQ operating system for the QL. This version comes with the equivalent of the pointer environment files PTR_GEN, WMAN and HOT_REXT built in and offers a large number of additional features over SMSQ, e.g. additional devices and device features, device buffering and the ability to change display resolution
SS	Single Sided, Refers to a type of floppy disk, or its drive
SSS	The QL Sampled Sound System. Enhanced sound system used by Amiga QL emulator, Q40, Q60 and QPC2
Stack	A sort of list where values are stored in a pile of numbers, and values can only be added to or taken away from the end of the stack. Often this is just a temporary storage area fro a number, from where it can be recovered later. Computers may have a system stack exclusively used by the computer's operating system, and a user stack exclusively used by the user's program.
Stipple	Although a standard QL has only 4 or 8 colours to display, you can use a combination of pixels next to each other with two different colours which look a bit like a third colour, e.g. lots of adjacent red and white pixels might look like a shade of pink. Sometimes referred to as a pattern of colours.
String	A data structure which holds text, e.g. LET a$ = "Hello"

Structured Programming	The art of writing programs which do not refer to line numbers. In other words, in SuperBASIC, programs which do not use keywords like GOTO and GOSUB
SuperBASIC	The QL version of BASIC, this was designed by a lady called Jan Jones for Sinclair
SVGA	Super Video Graphics Array. A PC graphics card. On the QXL or QPC an SVGA mode implies a display of size 800 pixels across and 600 down
SW	Shareware, a method of software distribution where the author lets you use either a cut-down version of a program, or a time-limited piece of software. If you like and wish to continue to use the full version of the software, you are expected to contact the author and pay a fee, for which you'll sometimes get a non-limited version of the software and product support. Sometimes used as an abbreviation for 'software' only
Syntax	Rules about the correctness of a part of a program. This is pretty much the same as the meaning of the word when used in conjunction with spoken language (nouns, verbs, etc). Syntax defines how a part of a program should be written, e.g. the syntax of a LET command such as LET a$ = 'Hello' dictates that the word LET is followed by a variable name, then an equals symbol, then a value to be assigned to that variable. Think of it as the grammar of a command or function in a program.

Programming in Assembler, Part 33 LibGen - Library Generator - Part 4

by Norman Dunbar

Introduction

In the last issue, we ended up with an application that was getting somewhere. But it's not finished yet. This issue we will look at what happens when the user requests that we load a symbol file. In this article, as before, we shall add code gradually, starting simple and getting more complicated as we go on.

Gwasl Update

George has updated gwasl to version 2.05 to fix a slight bug in error handling. To be sure you have the latest version, please download it from http://gwiltprogs.info/gwaslp08.zip. We shall be using this version from now on.

Errata

In the previous edition, Volume 17 Issue 2 dated December 2012 to February 2013, George Gwilt spotted a couple of errors in my code.

* In the IW_PRINT sub-routine, at label IWP_PRINT, we have the following:

```
iwp_prnt moveq #io_sstrg,d0
         move.w (a3)+,d2
         beq.s iwp_exit
```

The code is incorrect in that it introduces a bug. If there is no string to print, the code exits directly to iwp_exit where D0 is tested and the condition codes set. Unfortunately, this means that the condition codes will be set to reflect the fact that D0 is currently sitting with the value #IO_SSTRG and not an error code. This is a bad thing!

The code above must be changed as follows:

```
iwp_prnt move.w (a3)+,d2
         beq.s iwp_exit
         moveq #io_sstrg,d0
```

In this corrected version, D0 will contain zero on return from the previous trap and thus, on early exit to IWP_EXIT, the condition codes will be set to reflect this zero value, and the code will assume that all is well, which it is. The window has been cleared and nothing has been printed – which is exactly what is desired.

* Also in the sub-routine IW_INPUT, on page 22, I have a comment near the bottom of the page that details a bug in the documentation for the WM_ENAME vector. The comment is incorrect and should be removed. This has arisen due to a misunderstanding of the documentation, on my part.

I assumed that when the docs mentioned that the condition codes would be set, that what they really meant was that D0 would be set. A completely incorrect assumption on my part. However, I wrote the library routine with that assumption in mind. As it turned out, after an exchange of emails with George, the manual is correct and I am wrong.

If you leave the code the way it is, D0 will be set to negative, zero or positive as I originally misunderstood the manual to say. It does no harm, but the comment itself is wrong and should be removed as there is not a bug in the manual.

This also means that on page 21, the first line of the note, where it says that 'D0 will be zero if ENTER was pressed...to terminate the edit.', it is wrong. That whole paragraph can simply be ignored. Typpex it out if you must! ;-)

My apologies for these problems, and my thanks to George for his assistance in resolving my issues.

* George also pointed out that in the routines CP_STRING and AP_STRING, where the test for a zero length string is made, it is not necessary as the DBF instruction will detect this and do nothing anyway. What George says is true, but I tend to like to be explicit in my code. To this end, you can, if you wish to save a couple of bytes and clock cycles, remove the BEQ.S CS_EXIT and/or BEQ.S AS_EXIT instructions from near the start of these routines.

* George further pointed out my use of the BLT instruction as opposed to the BMI command. Once I checked my documentation, I discovered that George is correct – although the conditions under which BLT wouldn't work are unlikely, I have decided to correct the code. There are three occasions where the instruction BLT.S xx_EXIT, where xx is a two letter prefix, should be changed to BMI.S xx_exit instead. They three locations are in the routines at SYM_HIT, LIB_HIT and BIN_HIT. Please change all three BLT instructions to BMI.

What's the difference? Well, my code is looking for a negative number, and if found, currently bales out via the BLT instruction. How do you determine whether a number is negative or not? The N flag tells you. If the N flag is set, the number in question is negative, otherwise, it is positive.

The BMI instruction checks the N flag and branches accordingly. The BLT instruction, on the other hand, will branch only if the N flag is set and the V (overflow) flag is not set, or, if the V flag is set and the N flag is not.

* And finally, here's a couple I spotted myself. Unfortunately, because I'm running on QPC with a virtual) 68020, the following two bugs don't cause an exception, however, running on a standard QL, they would.

The code at LI_UNAV and LI_AVAIL currently looks like this:

```
li_unav   move.l #$11111111,ws_litem+li_libfile(a1)
          bra.s li_rdrw
...
li_avail move.l #$01011101,ws_litem+li_libfile(a1)
```

```
li_rdrw  moveq #-1,d3
         jmp wm_ldraw(a2)
```

The problem is, LI_LIBFILE is an odd value - it's $03. Although A1.L and WS_LITEM are both guaranteed to be even, the effective address of the MOVE.L is an odd address. As I mentioned, this works perfectly well on QPC with its 68020 processor, but the 68008 in our bare bones QLs will raise an address exception. We need to change the code as follows:

```
li_unav  move.b #wsi_mkun,ws_litem+li_libfile(a1)  Make Libfile unavailable
         move.w #$1111,ws_litem+li_load(a1)        Make Load and Save unavailable
         move.b #wsi_mkun,ws_litem+li_binfile(a1)  Make Binfile unavailable
         bra.s li_rdrw
...
li_avail move.b #wsi_mkav,ws_litem+li_libfile(a1)  Make Libfile available
         move.b #wsi_mkav,ws_litem+li_load(a1)     Make Load available, Save
stays Unavailable
         move.b #wsi_mkav,ws_litem+li_binfile(a1)  Make Binfile available

li_rdrw  moveq #-1,d3
         jmp wm_ldraw(a2)
```

Another Library

The processing for the 'Load' and 'Save' loose items requires a bit of file processing. I have a library for this, as you might expect. So, without any further ado, here it is. Mine is called win1_source_qltoday_openfiles_asm at the moment, and is simply included as a plain source file, it's not a proper library yet. If you save your file in a different place, please be sure to change the name in the 'in' command.

```
;======================================================================
; This file contains useful utilities for opening files etc.
; It's just crying out to become a library! ;-)
;----------------------------------------------------------------------
; F_OPEN      - Open a file. Old Exclusive mode.
; F_OPEN_IN   - Open a file for input. Old Shared mode.
; F_OPEN_NEW  - Open a new file. New Exclusive mode.
; F_OPEN_OVER - Open overwrite. New Overwrite mode.
; F_OPEN_DIR  - Open a directory. Open Directory mode.
; F_CLOSE     - Close any open file.
; F_SYNC      - Flush buffers to device.
; F_POS_ABS   - Set absolute file position.
; F_POS_REL   - Set relative file position.
;======================================================================
; Entry Registers.
;
; Unless otherwise specified, all routines use the following registers
; on entry:
;
; A0.L Pointer to QDOSSMQ string defining the file name.
;----------------------------------------------------------------------
; Exit Registers.
;
; Unless otherwise specified, all routines exit with the following
; register usage. Any register not listed will be preserved by the
; routines.
;
; D0.L Error code. Zero = no errors.
; A0.L Channel id.
; Flags - Z set to reflect the value in D0.
;======================================================================
io_openx      equ 0   SuperBasic OPEN
io_open_in    equ 1   SuperBasic OPEN_IN
io_open_new   equ 2   SuperBasic OPEN_NEW
io_open_over  equ 3   SuperBasic OPEN_OVER
io_open_dir   equ 4   No SuperBasic equivalent.

;=====================================================================
; F_OPEN - Open an existing file.
;=====================================================================
f_open
        move.l d3,-(a7)         Save D3
        moveq #io_openx,d3      IO Open code
        bra.s f_open_all        Go do it, and exit.

;=====================================================================
; F_OPEN_IN - Open a file for input.
;=====================================================================
f_open_in
        move.l d3,-(a7)
        moveq #io_open_in,d3
        bra.s f_open_all

;=====================================================================
; F_OPEN_NEW - Create a new file.
;=====================================================================
f_open_new
        move.l d3,-(a7)
        moveq #io_open_new,d3
        bra.s f_open_all

;=====================================================================
; F_OPEN_OVER - Open a file, overwriting existing contents.
;=====================================================================
f_open_over
        move.l d3,-(a7)
        moveq #io_open_over,d3
        bra.s f_open_all

;=====================================================================
; F_OPEN_DIR - Open a device directory.
;=====================================================================
f_open_dir
        move.l d3,-(a7)
        moveq #io_open_dir,d3

;=====================================================================
; F_OPEN_ALL - general purpose working "open" routine.
;=====================================================================
f_open_all
        movem.l d1-d2/a1,-(a7) Save working registers
        moveq #io_open,d0       IO_OPEN trap code
        moveq #-1,d1            File is for this job
;                               D3 already set
;                               A0 already set
        trap #2                 Do it
        movem.l (a7)+,d1-d2/a1 Restore working registers
        move.l (a7)+,d3         Plus D3 from above
        tst.l d0                Set Z flag
        rts                     Done
```

The first few routines are all to do with opening files. All that they do is set the required open mode in D3 then branch down to doing all the hard work at f_open_all. D3 is also preserved by the individual routines as we need to keep it safe from changes.

On arrival at f_open_all, we preserve all the other working registers and call out to QDOSMSQ to open the file for us. On return, A0 holds the channel id and D0 any potential error codes. After restoring all the working registers, and D3, we set the flags according to D0's error code and exit.

```
;======================================================================
; F_CLOSE - Close any open file.
;======================================================================
; Entry Registers:
;
; A0.L Channel Id
;----------------------------------------------------------------------
; Exit Registers:
;
; D0.L Error code (channel not open)
; A0.L Corrupted.
; Flags Z set according to D0.
;======================================================================
f_close
        moveq #io_close,d0      Close file trap code
        trap #2                 Do it
        tst.l d0                Set Z flag
        rts                     Done
```

The f_close routine simply closes the channel id passed in A0, and returns with any errors in D0 and the Z flag set accordingly.

```
;======================================================================
; F_POS_ABS - Set the current file position pointer to an absolute
;             position in the file.
;======================================================================
; Entry Registers:
;
; D1.L File position required.
; A0.L Channel Id.
;----------------------------------------------------------------------
; Exit Registers:
;
; D0.L Error code.
; D1.L New file position.
; Flags Z set according to D0.
;======================================================================
f_pos_abs
        moveq #fs_posab,d0      Abs position trap code
        bra.s f_pos_all         Do it

;======================================================================
; F_POS_REL - Set the file position to a new location relative to the
;             current location.
;======================================================================
; For register usages, see F_POS_ABS above.
;======================================================================
f_pos_rel
        moveq #fs_posre,d0      Relative position trap code

;======================================================================
; F_POS_ALL - General do it all file positioning routine.
;======================================================================
f_pos_all
        movem.l d3/a1,-(a7)     Preserve working registers
        moveq #-1,d3            Timeout
;                               D0 already set to trap code
;                               D1 already set to file location
;                               A0 already set to channel id
        trap #3                 Do it
        movem.l (a7)+,d3/a1     Restore registers
        tst.l d0                Set Z flag
        rts                     Done

;======================================================================
; F_FLUSH - Flush a file's slave blocks to disc.
;======================================================================
; Entry Registers:
;
; A0.L Channel Id.
;----------------------------------------------------------------------
; Exit Registers:
;
; D0.L Error code.
; Flags Z set according to D0.
;======================================================================
f_flush
        movem.l d1/d3/a1,-(a7) Preserve working registers
        moveq #fs_flush,d0     Trap code
        moveq #-1,d3           Timeout
;                              A0 already set to channel id
        trap #3                Do it
        movem.l (a7)+,d1/d3/a1 Restore registers
        tst.l d0               Set flags
        rts                    Done
```

The above routines carry out useful file flushing and positioning facilities. As ever, the channel id is passed in A0, and for the file positioning routines, a file location in D1.L. After preserving the working registers, the routines simply call out to QDOSMSQ to do the hard work before restoring any working registers and exiting with the Z flag set according to the error code in D0.

The following one line needs to be added to the end of libgen_asm to include the above library of useful file handling routines.

```
        in    win1_source_qltoday_openfiles_asm
```

It should be added after the existing line that includes my pe_utilities_asm source file.

Load Processing

When the user hits the 'Load' loose item, the file named as the Symbol file, is opened for reading. It is then processed in two passes.

The first pass simply counts the number of code offset lines that will be added to the menu. This is done by looking along the entire line of text read in, for the appearance of the characters '*+' as these two indicate that the entry is for a code offset rather than an equate. If we find what we are looking for so we increment the counter. Any errors cause an immediate return with the Z flag indicating success or failure.

The second pass resets the file position back to the start, and begins reading again. This time, for each entry that represents a code offset, we extract the label name from the entry, and add it to the buffer allocated for the application menu. Again, in the event of an error, we return with the Z flag holding the success or failure status.

If all is well, then at end of file, the file will be closed and the buffer added to the application sub-window as a menu. All items in the menu will be selected by default. If the file loads correctly, the 'Save' loose item will be enabled and the 'Load' loose item will be reset to available.

I did consider reading the _sym file directly and bypassing the need to create a _sym_lst file. I got as far as diagnosing the internal structure of a gwasl _sym file and even worked out a fairly simple manner of processing the load requirements. Unfortunately, George advised that I stick with the _sym_lst file as it is identical whether the original _sym file was created with gwasl or gwass - as these produce a different _sym file format.

I did a quick test using gwass instead of gwasl and sure enough, the _sym files are different. Equally, the _sym_lst files are also different, so I need to scan each line read in from the file looking for the special characters I want instead of just checking a specific location in the line. Never mind, this way, if George changes things, LibGen will still work.

LibGen Code

As with the other hit routines, we start off quite simple. The following code is all that is required as the main 'Load' loose item hit routine:

```
;---------------------------------------------------------------------
; LOAD loose item action routine.
;---------------------------------------------------------------------
afun0_4  movem.l d5-d7/a0-a4,-(a7) Preserve important registers.
         bsr load_hit             Do it all.
         movem.l (a7)+,d5-d7/a0-a4 Restore important registers.
         move.w #$0101,ws_litem+li_load(a1) Only works because li_load is even!
         bra li_rdrw              Load = available, Save = available.

;---------------------------------------------------------------------
; This code carries out all the nasty work for a hit on the Load
; loose item. It is called from afun0_4 above.
;---------------------------------------------------------------------
load_hit bsr.s ld_pass_1          Count the entries for the menu
         bne.s ld_exit            Oops, an error occurred
         bsr ld_pass_2            Build the menu

ld_exit  rts                      Done, D0 = error or not

;---------------------------------------------------------------------
; Load file, pass 1, open the symbol file and count the code offsets.
;---------------------------------------------------------------------
ld_pass_1
         moveq #0,d0
         rts

;---------------------------------------------------------------------
; Load file, pass 2, build the menu, close the symbol file.
;---------------------------------------------------------------------
ld_pass_2
         moveq #0,d0
         rts
```

The main meat of the hit code is yet to be written, however, assembling the above and executing it shows that when the 'Load' loose item is hit, both it and the 'Save' one become available. Nothing will happen if you now hit or do the 'Save' loose item because it still has a dummy hit routine that does nothing, other than to reset the loose item to available. If you have a slow QL, you might see the 'Save' loose item flash as it has its status reset from selected to available.

An important point to note. The setting of the two loose items to available by a single MOVE.W instruction only works because ws_litem is defined as even. its value is $40 and, as I already know that li_load is also even, then the MOVE.W works. If either of these (or for that matter, A1) were odd, there would be an address exception on a standard QL, but not on QPC. If you ever wish to add extra loose items to this utility, and want to be extra safe, then change the single MOVE.W above, to the following:

```
        move.b #wsi_mkav,ws_litem+li_load(a1) Make Load available.
        move.b #wsi_mkav,ws_litem+li_save(a1) Make Save available.
```

We need to add a couple of helper routines next, and then, we can hook them up to ld_pass_1 replace the dummy code we have there at present.

As we have to scan each line of the _sym_lst file for the characters '*+' in both passes, it's probably wise to extract the code to a separate sub-routine. However, there's a slight problem. In the first pass we are simply counting entries so that we can allocate the correct amount of memory to build the menu, while in pass 2 we have to extract all the names and offsets to actually use in the menu. What to do?

The answer is simple, if we set an address register to the address of a sub-routine to carry out the required work, we can use the JSR (An) instruction each time we find the required text. That way, each pass will set the register to a different address and whatever code is at that address when we find a line containing a code offset, will be executed. Simple!

The following code is our equivalent of INSTR. You should type it in just above where ld_pass_2 is currently located.

```
;=====================================================================
; LD_INSTR - scans each line of the buffer at (A1) looking for the
;            characters '*+' and if found, calls the code at the
;            address in A2.
;=====================================================================
; Entry Registers:
;
; A1.L Buffer to search. Normal QDOS format of word and bytes.
; A2.L Address of code to be called on a "find". Zero if nothing.
;---------------------------------------------------------------------
; Exit Registers:
;
; A1.L Address in buffer of the character after the '*' we looked for,
;      if found. Points at the '+'. If not found, one past the buffer
;      end if not found.
;
; D0.L  0 = Found the '*+'.
;      -1 = Not found.
;      -ve = Appropriate error.
;
; Flags Z set according to D0.
;=====================================================================
ld_instr moveq #0,d0                Need D0.L to be zero
         move.w (a1)+,d0            Grab the word count
         beq.s ldi_done             Unlikely, but you never know!
         bra.s ldi_next             Decrement before comparing

ldi_scan cmpi.b #'*',(a1)+          Compare and increment
ldi_next dbeq d0,ldi_scan           Try again or drop out

;---------------------------------------------------------------------
; If we get here we have either found a '*' or we haven't. If D0.W is
; -1 then we didn't find it. Otherwise the Z flag should be set.
;---------------------------------------------------------------------
         bne.s ldi_exit             Bale out, D0 = -1 = not found
         cmpi.b #'+',(a1)           Compare, but don't increment
         bne.s ldi_next             Keep searching (unlikely!)
```

```
;----------------------------------------------------------------
; Found '*+' if A2 is non-zero, call the code addressed in A2.
;----------------------------------------------------------------
        cmpa.l #0,a2              Anything to execute?
        beq.s ldi_exit            No, bale out D0 = 0 = found
        jsr (a2)                  Yes, call the code

;----------------------------------------------------------------
; A not found means we arrive here with D0.W = -1 instead of D0.L. We
; must extend D0.W to a long or the calling code will barf!
;----------------------------------------------------------------
ldi_exit ext.l d0                 Must exit with error code in D0.L

ldi_done rts                      Done
```

As you can see it's quite simple. We are passed a buffer address in A1.L and a code address in A2.L. D0 is set to the word count for the size of the data in the buffer, and then we enter a loop to search for the required '*' character in the buffer. If we hit the end of the buffer, D0 will be -1 and the Z flag will not be set, so we exit directly to the RTS instruction at ldi_done which means we return -1 as a flag that we didn't find anything.

Note that in the event that we don't find the '*' we are looking for, D0.L holds the value $0000FFFF. The calling code adds 1 to D0.L to determine if it was a not found or some other error, so we have to make sure that D0.L holds $FFFFFFFF.

The code at ldi_exit does this, but also does it when we have found it, whether or not we call any code in A2, so in that case, it's an un-required instruction. However, it does nicely set the flags for us so we don't have to TST.L D0 on the way out.

If we do find an '*' character, we exit from the bottom of the loop with the condition codes set to 'eq' (the same condition as the 'cc' part of the Dbcc instruction) so we know we found it. A1.L is pointing at the character after the '*' and we hope that it is a '+'. If it is not a plus, then we skip back into the loop to see if there are any more '*' characters. This is unlikely in a _sym_lst file, but you never can be too sure!

Assuming the character at A1.L is a plus, we check if A2.L is zero and of so, exit setting D0 to zero to indicate found. If A2.L holds some other value, we call the code pointed to by A2.L and, on return, make sure the flags are set according to any potential error codes in D0, and exit.

We are now ready to write the code to replace the dummy code at ld_pass_1.

```
ld_pass_1
          lea sym_buffer,a0       File name
          bsr f_open_in           File must exist already for reading
          bne sui                 Error exit, dramatic and fatal!

          moveq #0,d7             Counter register
          lea ld_p1_count,a2      Call back code to do the counting

p1_entry  moveq #128,d2           Size of buffer space. (Preserved)
          moveq #-1,d3            Timeout (Preserved)
```

The main entry point to pass attempts to open the file who's name is currently stored in the sym file buffer. If it fails to open for any reason, I'm afraid the application simply dies. This is not ideal and in a 'commercial' quality application this would simply be unacceptable. I only do it here to reduce space in the magazine.

Note: I do know how irritating it can be for a program to just vanish. During testing I had a bug that caused the program to quit whenever a line didn't contain an asterisk. The reason was that D0.L held $0000FFFF and not $FFFFFFFF on exit from ld_instr.

The next step is to zero D7 which is acting as our counter. The whole of D7.L is set to zero although the code only requires the low word. A2.L is loaded with the address of the callback routine that does the counting for us. Every time there's a hit in the data read from the _sym_lst file, we jump to the code whose address is held in A2.L. On this pass, it simply counts. (See below.)

P1_entry is the second entry point to this code. There's no point duplicating huge chunks of code, and as pass 1 and pass 2 are both required to read the file, and scan each line of text read from it, we can use the code in pass 1 as part of pass 2. P1_entry will be where pass 2 joins in with the fun.

At this point, we have everything initialised for pass 1. The buffer length for IO_FLINE is set as is the timeout. These two registers are preserved through the calls to IO_FLINE so can be initialised once only, prior to the start of the main loop for each pass.

```
p1_loop  lea p1_buffer+2,a1         We need to reinitialise the each time
;                                   A0.L already has the channel id
;                                   D2.W already has the buffer size
;                                   D3 already holds the timeout
         moveq #io_fline,d0         The trap code we desire
         trap #3                    Read up to 128 bytes ending with a lf
         tst.l d0                   Errors?
         beq.s p1_not_eof           No, all ok, skip

p1_eof   cmpi.l #-10,d0             Was it EOF?
         bne sui                    No, error, die horribly!
         bra.s p1_exit              Must be EOF, all done for pass 1

p1_not_eof
         subq.w #1,d1               Lose the terminating lf character
         beq.s p1_next              Nothing to do here, empty line
         lea p1_buffer,a1           Buffer address again
         move.w d1,(a1)             Make into QDOSMSQ string
```

The first part of the main loop sets A1.L to the third character of the buffer we will use to read each line of data from the _sym_lst file into. IO_FLINE fetches bytes, not a QDOSMSQ string with a word count, so we cannot simply point A1 at the start of the buffer.

We call IO_FLINE to read some text up to a maximum of 128 bytes, but normally terminating with a linefeed character. On return, if D0.L is EOF we exit as this concludes pass 1. If D0.L is holding any other error code, the application simply takes the easy way out and dies horribly.

The number of characters read, including the linefeed is returned in D1.W from which, we subtract 1 to lose the trailing linefeed. We skip to the end of the loop if the length is now zero as we can't scan an empty string. If there is work to do, we reset A1.L to the word count at the start of the buffer and store the length word, now not including the linefeed. We are now ready to scan the fresh data in the buffer.

```
         bsr.s ld_instr             Scan for the required characters
         beq.s p1_next              We found it, do next line of data
         addq.l #1,d0               Minus 1 was not found
         bne sui                    Oh dear, some other error. Die

p1_next  bra.s p1_loop              Let's go round again

p1_exit  moveq #0,d0                No errors. We come here from EOF only
         rts
```

The above code calls the 'instr' code to look for the two desired characters. If found, D0.L will be zero on return. In that case, we skip to the bottom of the main loop ready to go round again. If D0.Lholds minus 1 then that indicates not found, which isn't an error, so we will also go around again until we hit EOF. Any other errors cause the application to die horribly as usual. (I really don't like that!)

```
p1_buffer
        ds.w 65                         128 bytes + a word count
```

The code above reserves a 65 word buffer which allows for 128 characters and a word count. This will be ample for our needs.

We must also write the call back routine that does the pass 1 counting.

```
;=====================================================================
; LD_P1_COUNT - called from LD_INSTR during pass 1 when we find the
;               required '*+' characters in the buffer. All we do is
;               increment a counter, in D7.W.
;=====================================================================
; Entry Registers:
;
; D7.W Counter of hits so far.
; A1.L Pointer into search buffer. Points at the '+' character.
; A2.L Address of ld_p1_counter.
;---------------------------------------------------------------------
; Exit Registers:
;
; D7.w Incremented by one from the entry value.
; D0.L Zero.
; Flags Z set.
;---------------------------------------------------------------------
ld_p1_count
        addq.w #1,d7                    Simply increment the counter
        moveq #0,d0                     No errors. Sets Z flag.
        rts
```

The code can be assembled and executed. You are now able to enter a sym file name, and load it. Well, you are able to carry out pass 1 of the load process which does nothing much except count the required entries we will need in the application sub-window menu.

Next time, we will add in pass 2 to build and display the menu.

I2C Interface for QL Emulators Part 7 (and now for non Minerva MK2 Black Box QL's)

by Ian Burkinshaw

Some of you may have been put off by the cost of the By-Vac BV4221-V2 since, at the time of writing, this device costs £24.50. Another down side of this device is that it will only work with PC based emulators since it uses the USB port. So is there another way? Yes there is, it is cheaper – at least the main hardware solution – and will work with 'Black Box' QL's. However you will have to construct it, program a PIC micro-processor and provide a 5VDC regulated supply. So there is more work involved. But this is a really good way of using that old QL you have hidden away. I am using one as a model railway layout controller. Without delving (at least too far) into the 'Black Box' itself. As to the BT connectors used on standard UK QL's please see my comments in QLT 17, Issue 2, page 9.

The cost of the main active components is as follows:

From Farnell		
PIC12F675I/P	9759018	£1.08
MAX232CPE+	9725172	£3.67
From Rapid		
PIC12F675I/P	73-3284	£0.84
MAX232CPE+	82-0148	£1.41
10uF 25V Cap x 4	11-3686'	£0.031 each
0.1uF Cap x 2	08-0235	£0.021 each

Depending where you purchase the components, and adding things like connectors, this converter can be built for around £10. The prices are correct at the time of writing, but there may be minimum order requirements. Please check carefully when ordering parts.

You need to have the skills to program PIC chips. There are several ways of doing this. For example the Microchip PICkit2 made by the manufactures of the PIC micro's themselves, costs £32.26 from Farnell. An alternative is the Velleman kit (K8048) at £34.99 from Maplin. The Velleman kit is limited in the range of PIC devices it can program. However it will program the PIC12F675 as used in this converter and also the PIC16F84 as used in the PS2 mouse to game port converter by flyer.gio from Italy (see my review in QLT 17, Issue 2).. The Velleman kit is very good if you want to learn more about PIC's, since it is a programmer and development board, so you takes your choice. I have both these devices and they work very well. This does add to the overall cost, but is a good investment for other PIC based projects and to learn more about PIC devices.

There is a third (the cheapest) solution called WIN PIC. This is a PC based programmer, like the ones above, but with fairly simple hardware which you can make on some strip board. WIN PIC will program a wide variety of PIC devices, including the PIC16F84A for the PS Mouse project and the PIC12F675 for this project. Use the circuit diagram at
http://www.qsl.net/dl4yhf/winpic/index.htm#eprom_prog.

Since this has a better arrangement for the all important programming voltage. The power for the programming voltage can be provided with two (R22) PP3 batteries connected in series to give a voltage of about 18 volts, the on board regulator then reduces this to the required 12.7 volts for programming the PIC's. All the PC software and information for this programmer can be found at http://www.qsl.net/dl4yhf/winpicpr.html. Scroll down the page until the WIN PIC manual for the user manual. You will find the software download further down the opening page.

I have tested the WIN PIC programmer on both the 'PS Mouse' and the 'RS232 to I2C' projects and it works with no problem.

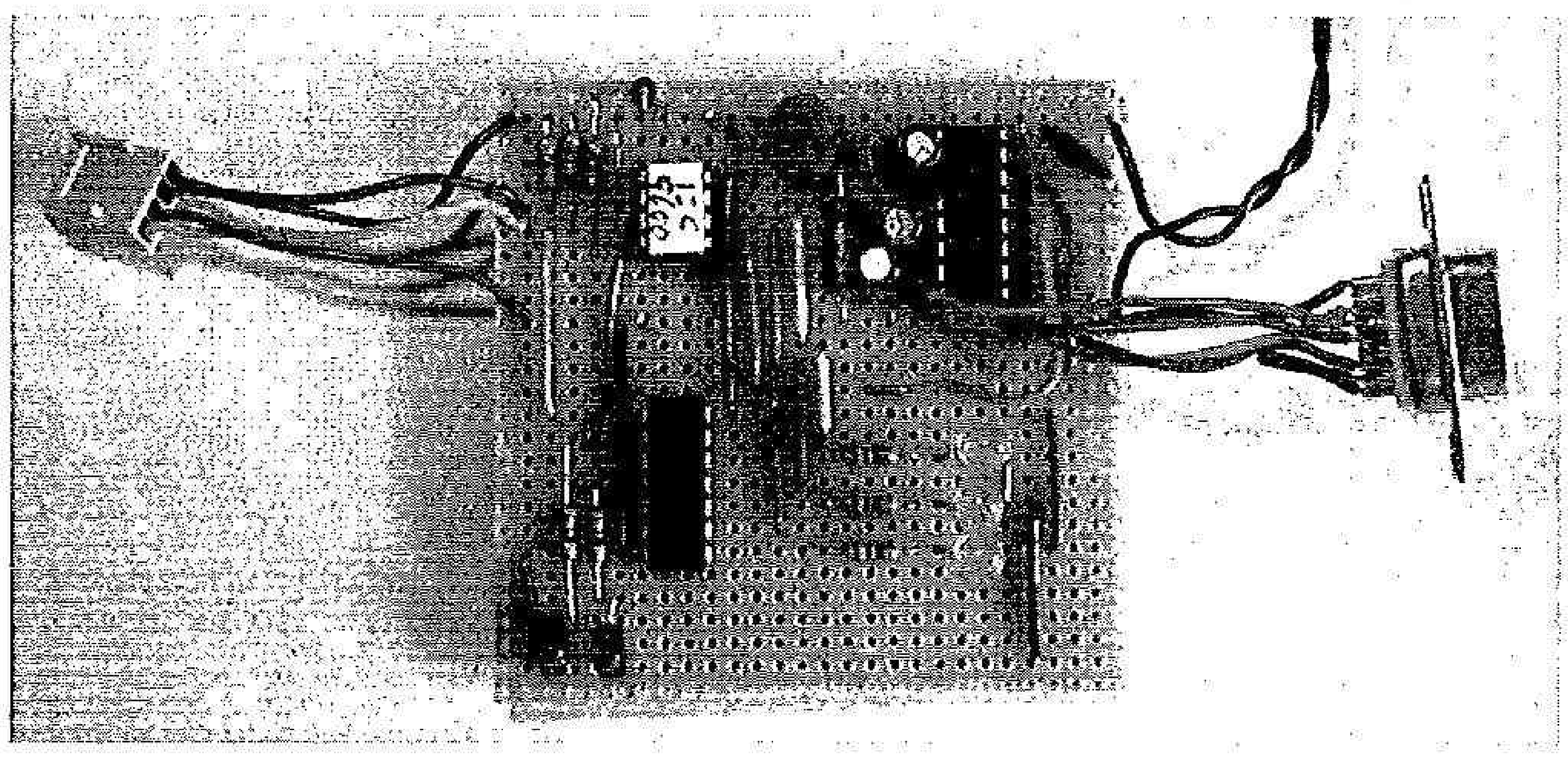
You will also need Microchips MPLAB IDE development tool, this is free to download from Microchips web site, please see below for link. Be warned this'is a big download.

In essence this PIC based converter is a RS232 to I2C converter. To use it with a PC based emulators such as QPC you will need either a RS232 on your PC or a USB to RS232 converter. These are available from such places as Maplin's. You may already have one of these. So again this may add to the cost. So the ByVac product could still be the best answer for you, if you don't have a USB to RS232 converter or PIC programmer.

So to the RS232 to I2C converter itself. The hardware and firmware for this project was developed by Andrew M Bishop. He is, as far as I know, not a QL user. This project was originally designed for Linux and PC based projects. See his web site for more information. You will also find other RS232 based converters as well, such as for PS2, SPI and Infra Red device. Which you may find of interest, but I will only be dealing with the I2C this time.

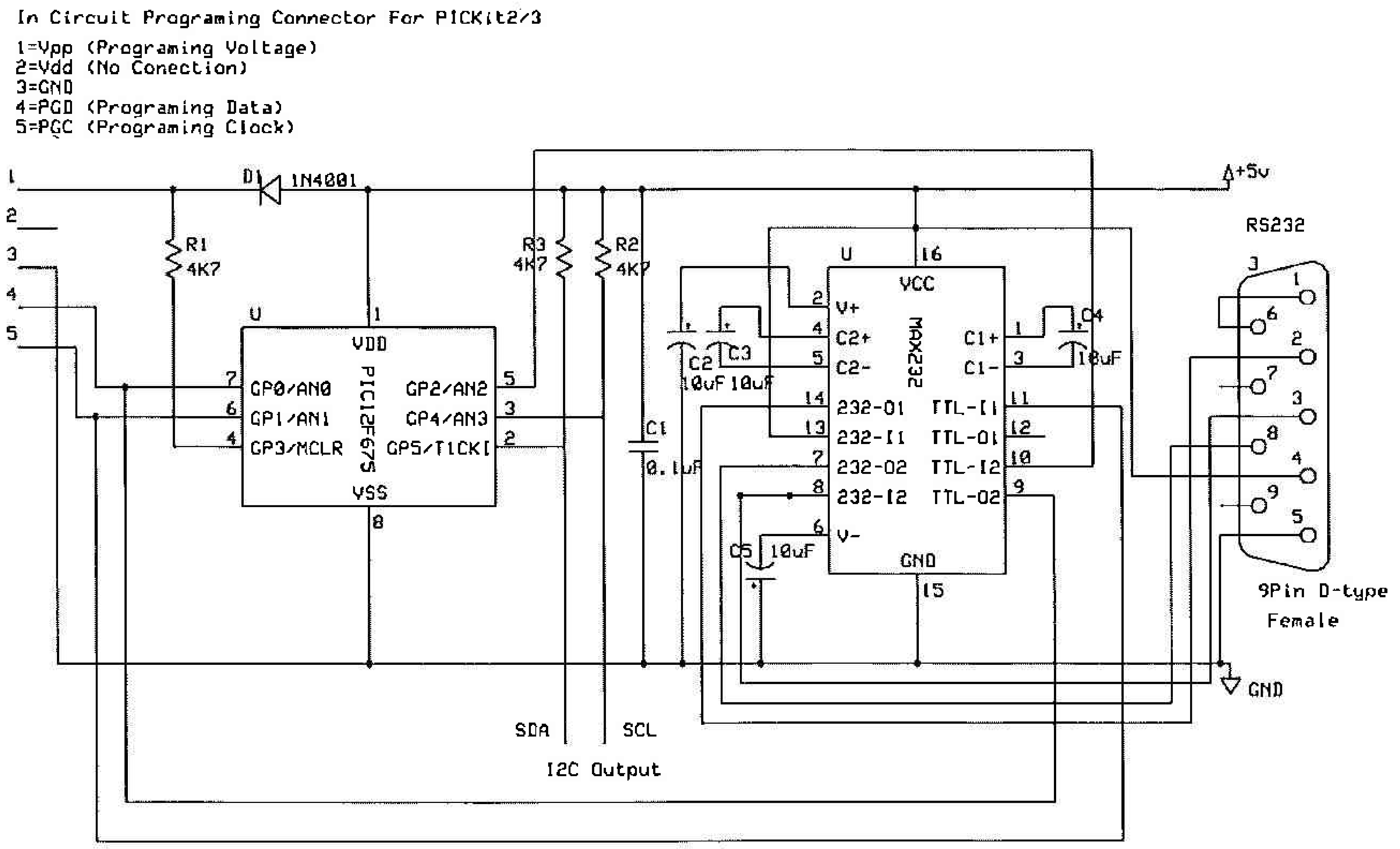
The nice thing about Andrew's projects is that all the source code as well at the compiled hex files are available in his amb-pic-code file. See below for the link for this. So you can see how these things work and make any changes you want for your own purposes.

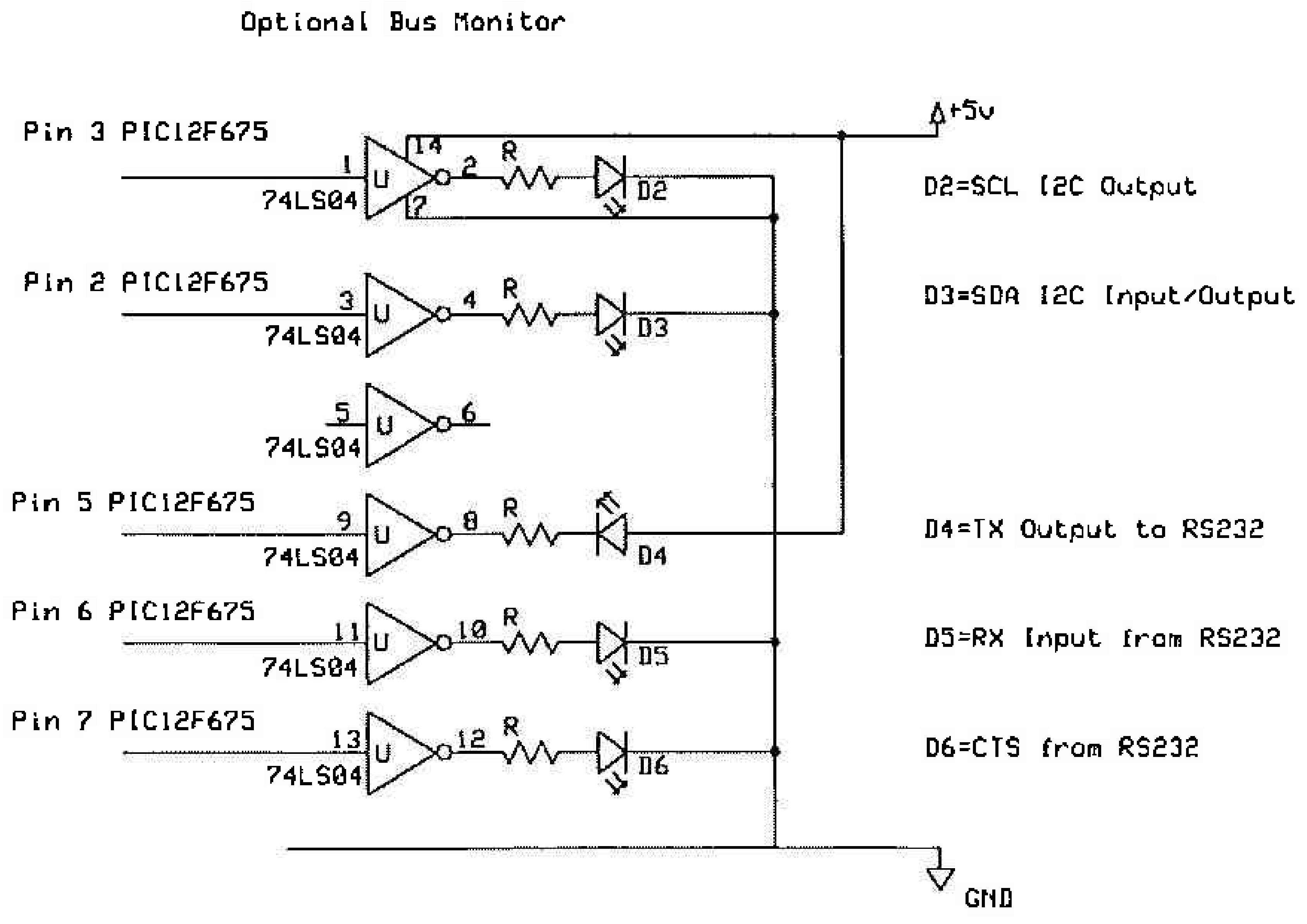
Below, an image of the RS232 to I2C converter



In its original form the converter works at 38400 baud, which is fine for emulator users like QPC2, but is far to fast for 'Black Box' QL users. So I have modified Andrews code to work at 9600 baud. I will explain how you change Andrews code.

The circuit diagram below shows the complete project, it breaks down into three parts. The PIC which does all the clever stuff and does the conversion work. This is the top left integrated circuit with just the PIC and three resistors and a diode. For PICKit2/3 users I have shown the connections for the programmer. Second is the MAX232 which is a level converter to change the +5v signals to and from the PIC to the +/- 8 volts required for the RS232 interface, this is the integrated circuit and associated components, top right. The third and final part is a LED monitor, which shows the traffic on all the I2C and RS232 lines, this is optional. I just put this in for my own testing purposes. For this project to work you just need the PIC and MAX232 parts of the circuit in the top diagram below. Unlike the BV4221 which received its power from the USB connection, you need a regulated 5VDC supply to this circuit.





As you can see it is very simple, with a low component count. There are not even any crystals, since the PIC is programmed to use an internal oscillator.

Using the original hex file **program.hex** from Andrew's library, which you will find in the amb-pic-code-2010-09-19/rs232-i2c directory. Then just follow the instruction for your PIC programmer. This will give a working PIC converter that works at 38400 baud.

This high baud rate is fine if you just want to use this converter with QPC2. However this is far to fast for a 'Black Box QL'. The maximum speed with a standard QL is 9600 baud. Now to change the baud rate from 38400 to 9600 you will need the MPLAB IDE. So download this from the Microchip web site and install it on your PC. No choice here has to be a PC.

Now to make things a little easier to understand I put all the required files to compile the PIC code in one directory. They are listed below you will find them in the amb-pic-code-2010-09-19/rs232-i2c directory and the amb-pic-code-2010-09-19/common directories.

```
Delay.inc
Delay_cycle.inc
Delay_ms.inc
Delay_us.inc
Eeprom.inc
Generic.inc
I2c.inc
I2c_basic.inc
I2c_master.inc
I2c_slave.inc
I2c_ssp.inc
I2c_ssp_basic.inc
I2c_ssp_master.inc
I2x_ssp_slave.inc
Reset_device.inc
Rs232.inc
Rs232_basic.inc
Rs232_rx.inc
Rs232_time.inc
Rs232_tx.inc
Program.asm
```

It is best to load the elements above into MPLAB IDE to edit and then save them. You could do this in Notepad, but the layout will not be very clear. Just go though all the above RS232 files and change any entry from 38400 to 9600. Recompile the code. If you do not want to do this then I have made available the hex file, which runs at 9600 to the editors of QLToday.

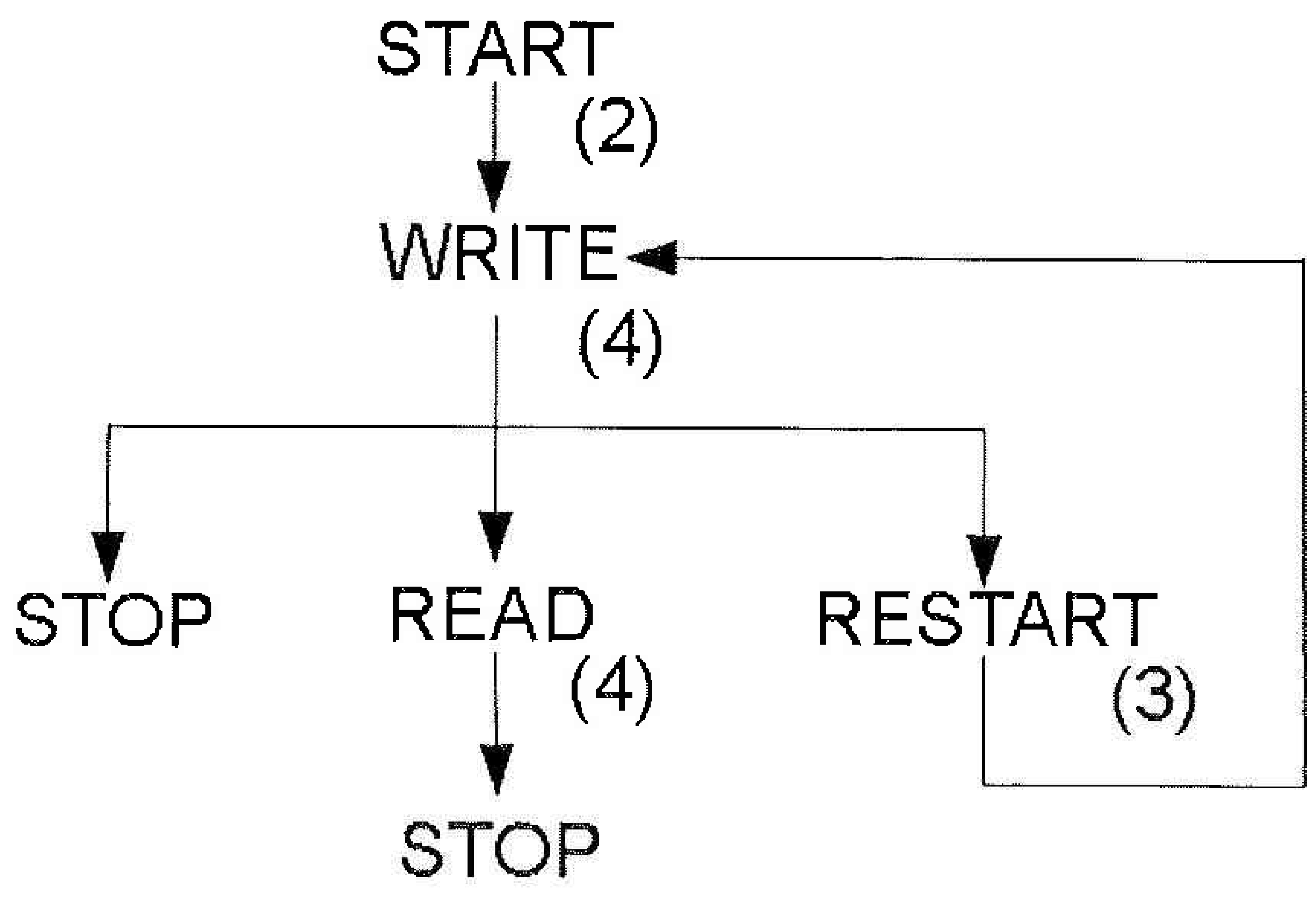
So that is the hardware now the software. The protocol for this converter is different from the BV4221 we have discussed during this series.

The commands are as follows:-

START	= 'S'	
STOP	= 'P'	
RESET	= 'R'	
WRITE COMMAND	= 'w'	
WRITE DATA	= 'x'	For use with EEPROMS
READ COMMAND	= 'r'	
READ DATA	= 's'	For use with EEPROMS

State machine that is implemented here.

(Brackets show error number if expected state is NOT reached)



State Machine

So the order of commands should look something like this :

```
'S'TART, write n bytes, STO'P'
```
or
```
'S'TART, write n bytes, 'R'ESTART, read m bytes, STO'P'
```

A simple example SB program is show below, which is fully commented, this can drive a PCF8574A in output mode :

```
10 REMark RS232 to I2C test
20 CLS
30 BAUD ser1,9600:REMark Sets the baud rate for the RS232 to I2C Converter, either
   9600 or 38400, depending on how the PIC has been programmed
40 OPEN#3;ser1:REMark opens the required serial port.
50 PRINT#3;CHR$(83);:REMark HEX='53',ASCII code='S' for START I2C command
60 PAUSE 10:REMark Delay to allow the PIC to process the data and be ready for the
   next character
70 PRINT#3;CHR$(119);:REMark HEX='77', ASCII code="w" write to I2C device
80 PAUSE 10
90 PRINT#3;CHR$(2);:REMark HEX='02' number of bytes to follow.
100 PAUSE 10
110 PRINT#3;CHR$(126);:REMark HEX='7E', ASCII='~' the I2C device address.
120 PAUSE 10
130 PRINT#3;CHR$(129);:REMark DATA to be sent to the I2C device (0 to 255).
140 PAUSE 10
150 PRINT#3;CHR$(80);:REMark HEX='50', ASCII='P' for the STOP I2C command
160 PAUSE 10
170 FOR a=0 TO 2: REMark This FOR NEXT loop reads the return which is sent after
    the STOP command has been sent to the RS232 to I2C converter.The return from
    RS232 to I2C converter should be 'OK' if all is well. This also where the
    errors will be returned if this go wrong, see State Machine above to help with
    debugging.
180 a$=INKEY$(#3)
190 IF a$="" THEN GO TO 180
200 PRINT a$;
210 NEXT a
220 CLOSE#3
```

I will leave it you to try reading data back from a device and amending my previous programs as you require.

I would like to thank Andrew M Bishop's help in the preparation of this article and his permission to feature his PIC development.

References

Andrew M Bishop's RS232 Converter page and software libraries
http://www.gedanken.org.uk/electronics/rs232-converters/i2c.html
http://www.gedanken.org.uk/electronics/amb-pic-code/

Microchip MPLAB IDE download site
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469

WIN PIC Programmer
http://www.qsl.net/dl4yhf/winpic/index.htm#eprom_prog
http://www.qsl.net/dl4yhf/winpicpr.html

DIY PS2 to game port converter by flyergio.
http://www.qlforum.co.uk/viewtopic.php?f=2&t=379
Software download for DIY PS2
http://www.mediafire.com/?xgmf9qx20sjxsbj

Last Minute News

QUANTA AGM

QL Today was unable to give details of the Quanta AGM in the main news section - the editor did not receive his copy of the December/ January Quanta Magazine.

The Workshop and AGM will be held in the usual venue in Manchester on 13th and 14th April. This is the third consecutive year the AGM has been held in Manchester, which offers the cheapest venue for a show with considerable savings on hiring costs and committee expenses. Last year the AGM cost Quanta £110, but a venue elsewhere would cost £500 to £1,000. In 2009 and 2010 Quanta held the AGM at central locations in the Midlands, but attendance was low.

This year Quanta is allowing electronic participation in the AGM via Skype in accordance with the new provision in the constitution. Earlier this year a committee meeting was held with one member participating via Skype.

Quanta is still having problems with members who formerly paid their subscription by standing order. Many have continued to pay at the old subscription rate and several have not replied to Quanta's letters on the matter. On advice from the auditor, their payments were temporarily placed in a suspense account before being treated as donations to Quanta. The result is a sharp, but not unexpected, fall in membership from 175 to 151. However there has been a sharp rise in the number of members opting to receive the electronic version of the magazine.

Quanta is redesigning its website following the transfer to a cheaper host.

SMSQmulator

A slight addition to the article on SMSQmulator on page 12.

When you reconfigure SMSQmulator to high colour mode, you may have to also increase the memory required to ensure that the QJewels program displays correctly.

SMALL ADS

Karl Illetschko from Austria is searching for German QUASAR magazines. He is rather flexible about how to obtain them - if somebody wants to get rid of them, he is happy to take them - pay for shipping and/or pay something for the magazines.

If somebody can lend him issues so that he can photocopy them, he'd be happy too.

Or, if somebody could send him scans or photocopies - that would be helpful as well.

Whichever option is possible, Karl is looking forward to it.

You can contact him via EMail:

outsider@netway.at

Jochens Last Words (for this issue)

Some time passed by since I called Geoff and discussed the situation. Time passed by since I wrote down the reason for the decision.

Now I am on page 53 ... wow, 54 pages total again although the calculation was based on 32 pages.

I have to admit it was a pleasure to lay it out ... very interesting stuff, as most of the time.

I am still missing some kind of 'relief' or 'stress burden taken from me' feeling ... probably because producing QL Today was nothing I would consider as 'pain' ... it has taken a lot of time, to produce (both the layout and the physical production until it was shipped). Which is very good in some way. Not that I will revert my decision - there are too many other factors which make it impossible to continue.

So let's prepare for the final issue. Once again, material is most welcome. We do not want to get 100's pages of praise (although we feel we did a good, reliable job over the 17 years) - let's try to produce a final, interesting issue at the normal quality level. If you have not written anything in the past, why not drop us a few lines now..?

So please remember:

Deadline for letters, articles, reviews etc. extended to 31st of May 2013

Issue 4 will not be shipped in June - it is scheduled for middle of September!

Have a great summer - we look forward to a great final issue, combined with a CD or DVD.

Your QL Team!